DOCUMENT RESUME

ED 022 504 LI 000 909

By-Artandi, Susan; Baxendale, Stanley
PROJECT MEDICO; MODEL EXPERIMENT IN DRUG INDEXING BY COMPUTER. FIRST PROGRESS REPORT.
Rutgers, The State Univ., New Brunswick, N.J. Graduate School of Library Service.
Spons Agency-Public Health Service (DHEW), Washington, D.C. National Library of Medicine.
Report No-PHS-LM-94
Pub Date Jan 68
Note-111p.
EDRS Price MF-$0.50 HC-$4.52
Descriptors-*AUTOMATION, COMPUTER PROGRAMS, COMPUTERS, *COMPUTER STORAGE DEVICES, DICTIONARIES, *INDEXING, *INFORMATION RETRIEVAL, *INFORMATION STORAGE, INPUT OUTPUT, OPERATIONS RESEARCH, SEARCH STRATEGIES
Identifiers-MEDICO, *Model Experiment in Drug Indexing by Computer

This report describes Phase 1 of a research project with the over-all objective of exploring the applicability of automatic methods in the indexing of drug information appearing in English natural language text. Two major phases of the research have been completed: (1) development of the automatic indexing method and its implementation on the test documents and (2) creation of a machine searchable file for the storage of the index records generated through automatic indexing. Indexing is based partly on a stored dictionary and partly on the identification of text characteristics which can be used to signal to the computer the presence of information to be indexed. The characteristics of the machine file were established with the desired search capabilities in mind. The file stores document references with their associated index terms. In addition to assigning index terms to documents the computer program also automatically assigns weights to the index terms to indicate their relative importance in the document description. The next two major phases of the work will involve the design and implementation of the search program and the evaluation of the automatic indexing method. Appended are sample program listings and output and operating instructions. (Author/JB)

LI 000909

ERIC/CLIS
AUG 5 '68
UNIVERSITY OF MINNESOTA

PROJECT MEDICO

FIRST PROGRESS REPORT

by
Susan Artandi
and
Stanley Baxendale

ED022504

Graduate School of Library Service
Rutgers, The State University
New Brunswick, New Jersey

January 1968

LI 000909

PROJECT MEDICO

(Model Experiment in Drug Indexing by Computer)

First Progress Report

(LM-94 Grant)

by
Susan Artandi
Graduate School of Library Service
Rutgers, The State University
and
Stanley Baxendale
Department of Computer Sciences
Rutgers, The State University

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

Graduate School of Library Service
Rutgers, The State University
New Brunswick, New Jersey

January 1968

FOREWORD

The research described in this first Progress Report was conducted under grant LM-94 from the Public Health Service National Library of Medicine.

Rutgers University personnel participating actively in the phase of the work reported here are:

Dr. Susan Artandi, Principal Investigator
Associate Professor, Graduate School of Library Service

Mr. Stanley Baxendale
Associate Professor, Department of Computer Sciences

Mrs. Ellen Altman
Mrs. Gillian McElroy
Graduate Students, Graduate School of Library Service

Mr. Charles W. Davis
Research Assistant, Department of Computer Sciences

A great deal of assistance and advice was received from Dr. Thomas H. Mott, Jr., Director, Center for Computer and Information Services and Chairman, Department of Computer Sciences, Rutgers University.

Extensive and valuable technical consultation was provided by Mr. Charles T. Meadow and Mr. Donald L. Dimitry, both of the I B M Corporation.

ABSTRACT

The objective of the research is to develop an automatic method for the indexing of drug information appearing in English natural language text. The indexing program automatically produces a computer file of document references with their associated index terms which can be searched on a coordinate basis for the retrieval of documents containing specified drug related information.

The method will describe a specific document with a degree of precision which makes it possible to retrieve information about a given drug regardless of the name that actually appears in the text of the document. In addition to assigning index terms to documents the computer program also automatically assigns weights to the terms to indicate their relative importance in the document description.

TABLE OF CONTENTS

Page

I. INTRODUCTION 1

II. STATUS OF THE PROJECT - SUMMARY 4

III. DEVELOPMENT OF THE AUTOMATIC INDEXING METHOD 6

Selection of Test Documents 6
Conversion of Text into Machine Readable Form 6
The Dictionary 7
Terms not in the Dictionary 9
Weights 10
Forms of Output 11

IV. DEVELOPMENT OF THE INDEXING PROGRAM 13

Hardware Considerations 13
Software Considerations 16
Overall Programming Approach 19
The Automatic Text Indexing Program 21
Dictionary Processing and Updating 27

V. APPENDIX 35

1. Sample Output of Automatic Indexing Using Full Text 37
2. Automatic Indexing Program Listing 55
3. Dictionary Processing Program Listing 67
4. Sorted Dictionary Terms with Linkages and Package Numbers 71
5. Type 1 and Type 2 Packages - Output 79
6. Coded Type 1 and Type 2 Packages 85
7. Sample Print-out of Transfer Address Matrix 89
8. Subroutine Program Listings 91
9. Operating Instructions 101
10. Rules for Keypunching Articles 107

I. INTRODUCTION

This is the First Progress Report on research in the automatic indexing of drug information conducted at Rutgers University under grant LM-94 from the Public Health Service National Library of Medicine.

The report describes Phase 1 of the research project whose over-all objective is to explore the applicability of automatic methods in the indexing of drug information appearing in English natural language text.

The relevance of this objective is based on two principal facts: the need to make available up-to-date and accurate information about drugs, and the desirability of minimizing human intellectual work in indexing.

The importance of making available up-to-date and accurate information about drugs is self-evident. The pharmaceutical industry has had a long term interest in readily accessible drug information, which is essential to survival in this highly competitive field. Recently, government agencies have stepped up their drug information activities to facilitate better control over the abuse of drugs, to support screening programs intended to explore the potential of various chemicals for their use as drugs, and to coordinate clinical experience regarding their adverse effects. The dissemination of drug information to the medical profession has also been of growing concern in recent years. This has led to attempts at the re-examination of communication methods in an environment in which a great deal of current information must be transmitted to a large and scattered group of users.

It is desirable to minimize human intellectual work in indexing because there is a scarcity of individuals who are qualified and willing to do the job and there is an increase in the number of publications which contain information that should be indexed.

Other reasons which gave impetus to research in the automatic indexing of natural language text relate to the greater availability and reduced cost of computers, the prospect of increased availability of text in machine readable form as a by-product of the printing process, and advances in research related to the development of character recognition devices.

Automatic indexing as referred to in this project means that the computer derives index entries from the natural language text of the document. This method should not be confused with mechanical storage and retrieval systems in which a machine searchable file is created from indexing done by humans.

The research reported on here used as a point of departure previous research done at Rutgers on automatic book indexing[1] and will also build on the experience and findings of others in the field.[2]

The automatic indexing method which is being developed in this project will describe a specific document with a degree of precision which makes it possible to retrieve information about a given drug regardless of the name that actually appears in the text of the document. Most drugs have three names: the chemical name which is a scientific and precise description of a chemical substance; the generic name or non-proprietary name, which is an abbreviated and frequently arbitrary version of the chemical name; and the trade (brand, proprietary) name which refers to a particular manufacturer's product. This implies an automatic capability to link a variety of trade names to a particular chemical composition and to the equivalent generic name. Display of these relationships and of other indications of document content in the index record can then be utilized in the mechanical searching of the automatically produced file.

An important part of the work concerns the testing and evaluation of the automatic indexing method. This requires the selection of methods for evaluation and the application of the evaluative criteria, taking into consideration the limitations imposed by the artificial controls of the experiment.

Evaluation is to include the comparison of indexing based on full text and on the abstracts of the same documents. This should provide interesting insights into such things as the role and potential usefulness of redundancy in language as compared to the use of "normalized" text.

---

[1] Artandi, S. Book indexing by computer. Ph.D. Thesis, New Brunswick, N. J. Rutgers, The State University, 1963.

[2] Stevens, M. E. Automatic indexing: a state-of-the-art report. NBS Monograph 91, Washington, D. C., National Bureau of Standards, 1965.

The scope of the project was defined through the number of documents indexed and through the number of drugs included in the particular group of drugs which was selected for the experiment. These limitations are not inherent in the indexing method. They are intended to limit the scope of the project in a manner that a minimum of expenditure should yield a maximum of findings from which generalizations can be made.

II. STATUS OF THE PROJECT - SUMMARY

Two major phases of the research have been completed:

1) Development of the automatic indexing method and its implementation on the test documents.
2) Creation of a machine searchable file for the storage of the index records generated through automatic indexing.

The test documents consist of English language periodical articles published in the medical literature and their subject matter relates to anticonvulsant drugs.

Indexing is based partly on a stored dictionary and partly on the identification of text characteristics which can be used to signal to the computer the presence of information to be indexed.

The characteristics of the machine file were established with the desired search capabilities in mind. The file stores document references with their associated index terms. In addition to assigning index terms to documents the computer program also automatically assigns weights to the index terms to indicate their relative importance in the document description.

As it stands now, the indexing program can detect information about anticonvulsants in the text and automatically index it in the following way:

1) index the drug under its preferred name--or names
2) compute and assign a weight to each term
3) assign its Chemical Abstracts registry number
4) assign the name of the chemical group to which it belongs
5) link modifiers to drug names to indicate the context in which they appear in the articles

Future Work

The next two major phases of the work will involve the design and implementation of the search program and the evaluation of the automatic indexing method.

The search program will be so designed that the primary information access points created through indexing can be further utilized through Boolean searches for the retrieval of more specific information. Through the use of the Boolean connectives AND, OR, NOT, the system will be able to answer search questions requiring the presence (or absence) of several parameters in specified combinations.

The evaluation of the indexing method will analyze the output of the indexing to ascertain the validity of the assumptions which formed the basis of the indexing method. This will include the evaluation of such things as the method of weighting, the basis for linking two terms in the index record, and the validity of the particular character string patterns used for the selection of index terms. Other evaluation work will relate to the comparison of outputs based on full text and on abstracts respectively, on the comparison with other systems, and on the ability of the system to deliver specified documents.

III. DESCRIPTION OF THE AUTOMATIC INDEXING METHOD

Selection of Test Documents

Published periodical articles were selected as the test documents for the experiment. For practical purposes it was necessary to define the scope of the experiment. This was done through the number of the test documents and through the number of drugs estimated to be contained in the drug group selected for the experiment. On the basis of the latter consideration anticonvulsants were selected as the experimental drug group. For a working definition those drugs were considered anticonvulsants which were classified as such in the major drug dictionaries and the open literature used in the compilation of the dictionary.

More specifically, for the selection of the test documents the definition used in their indexing by the National Library of Medicine was applied since they were selected from the output of a Medlars search for anticonvulsants limited to English language articles.

Since one of the objectives of the research is to compare indexing based on the abstracts of the same articles, abstracts of the selected articles were obtained or prepared when they were not published along with the article.

Conversion of Text into Machine-readable Form

There are essentially two major methods of obtaining machine-readable text for computer indexing:

1) as a by-product of the printing process; and
2) through some kind of conversion procedure using keyboard devices to produce cards or tape, or using optical scanning devices.

In the present experiment the text of the test articles was keypunched into cards with the text occupying columns 1 to 72. The only pre-coding that was done was to indicate the end of the bibliographic entry and the end of the article. The bibliographic entry consisting of author, title, and journal citation is followed by a sentinel card to separate it from the body of the article.

The keypunching was performed in such a way as to produce as straight-forward a representation of the text as possible. The constraints were the characters available on the keypunch. No attempt was made to code all representations of typed characters. For example, all versions of a letter, italic, upper case and lower case were represented by the letter without any added code to indicate the difference. Of course the use of punched paper tape instead of cards would have automatically taken care of the differentiation between upper and lower case letters. The end of sentences were indicated through the usual typographical practice of two spaces after the period and Greek letters were spelled out.

The Dictionary

Terms included in the dictionary differ by their nature and their function. Differentiated according to their nature we have the following types:

1) trade (brand, proprietary) names of individual drugs
2) chemical names of individual drugs
3) generic names of individual drugs
4) names of groups of chemical compounds
5) names of drug groups according to activity
6) terms which are not the names of drugs or chemicals

A hierarchical relationship exists among the "drug" terms which involves four levels. The most specific is the trade name (1); next the chemical and the generic name (2, 3); the next level is the chemical group name (4); and the fourth level is the group name indicating biological activity (5).

For example:

Level 1 - - - - - - - Mesantoin
Level 2 - - - - - - - 5-ethyl-3-methyl-5-phenylhydantoin
methylphenylhydantoin
Level 3 - - - - - - - hydantoins
Level 4 - - - - - - - anticonvulsants

Terms which are not names of drugs or chemicals (6) are used to modify, make more specific the drug terms. Examples of such terms are: activity, therapy, toxicity, etc. These are carefully linked in the index record, with the appropriate drug name so as to minimize the possibility of false drops. Linking is based on co-occurrence within a sentence and is marked by a slash between the terms in the printout.

According to function the dictionary includes those terms which are actually compared with the text to be indexed and those which appear in the index record.

Compilation of the dictionary for the project required the identification of those drugs which belong to the group of anticonvulsants and the selection of those non-drug terms judged to be of indexing value. It should be pointed out here that while a great deal of meticulous work went into the collection of the names of known anticonvulsants no claim is made for the list to be all inclusive. All inclusiveness as far as the drug list is concerned is of relatively minor importance in this case since the primary objective of the research is to demonstrate the feasibility of the indexing method. This, of course, will have to be taken into consideration in the evaluation of the indexing.

In addition to identifying anticonvulsant drugs it was also necessary to establish equivalencies among chemical, generic, and trade names. Since no single source exists which would make this information readily available it was put together from a number of different drug dictionaries and from the open literature. The most difficult part of this job was to locate the trade names of all drugs which have a common chemical composition. In some cases more than 30 trade names were traced to a single compound. Vocabulary control in the system is achieved through the establishment of equivalencies and through the arbitrary designation of terms to be those which will appear in the index record. This is essentially the control of synonyms which will be automatically taken into consideration by the search program.

While somewhat peripheral to the research reported on here, work related to the drug list demonstrated the great need for improved information sources which display the relationships among the generic, chemical and trade names of drugs. This seems especially important in view of the current controversy relating to the use of generic name drugs versus brand name products.

Index terms in the system may consist of one or several words and there is no limit set for their length. In this project the length of terms ranges between 4 and 65 characters.

With each drug name in the dictionary there is a "package" associated. A package consists of those terms which will appear in the index record whenever the original term appears in the text. Regardless of the number of times a term appears in the text its associated package will appear only once in the index record.

A package for chemical or generic names of drugs usually consists of the following:

Preferred chemical name

Preferred generic name

Chemical group name

Chemical Abstracts registry number

For a trade name the package consists of the same as above to which the trade name is added.

For terms which are not names of drugs the package consists of a single item, the preferred synonym or word form. The Chemical Abstracts registry number uniquely identifies specific compounds and serves as a unique machine address of that compound in the Chemical Abstracts Registry System. The registry number is a nine digit number which has no established pattern. Since it serves as a tag to identify all information associated with a given compound throughout the Chemical Abstract system its presence in the index record potentially can open up important additional sources of information for the user. The registry number also serves as an additional information access point.

Terms Not in the Dictionary

Inherent in dictionary-based indexing is the limitation of missing unknown new information, too new to be included in the dictionary. The problem is to devise some method by which a new anticonvulsant, for example, could be indexed when it is first reported on in the literature. "Catching" of new indexable information is also important from the point of view of dictionary up-dating. While it is possible to up-date the dictionary manually, it is desirable to make the process automatic.

The question, then, is how to alert the computer, in terms that it can understand, to the presence of information which should be indexed. At the current state-of-the-art of automatic indexing this problem should be restated as follows: how can you determine characteristics for strings of characters to indicate to the computer to take note of a particular string of characters. One way to do this is on the basis of location and this method is to some extent used in the assignment of modifiers discussed earlier. Another method is to base recognition on the pattern of the strings of characters. This second approach assumes previous knowledge of the string pattern of words which are usually associated with useful index terms. In this particular research the objective was to determine some common characteristics which names of drugs have, characteristics which will sufficiently distinguish drug names from other terms. Some characteristics which have been identified in case of drug names are such things as their length; an alternating string pattern of numbers, letters, and dashes; the capitalization of registered trade names followed by a capital R (words which begin and end with an upper case letter); the presence of such words as ethyl, methyl, etc., or the presence of Greek letters in chemical names.

The indexing program selects terms on the basis of their length to complement the dictionary method. Strings of characters exceeding 18 characters and not contained in the dictionary are put out in the form of a deck of cards for visual inspection. Those terms which are judged to be useful index terms are used in two ways:

1) put in the index record of the document which generated them
2) are used to update the dictionary

Evaluation of the output should shed some light on the usefulness of this man/machine method.

The question of defining appropriate string patterns for computer recognition is an important one. While there may not be an opportunity to explore it sufficiently in this project it deserves careful attention. The problem is particularly relevant to indexing in chemistry where one deals with a relatively standardized vocabulary as compared to the humanities or the social sciences, for example.

Weights

A considerable segment of experimental work in automatic indexing has evolved from H. P. Luhn's idea that the number of occurrences of a term in text can serve as clues to the importance of the concepts they represent. Thus text words can be designated as index terms on the basis of their frequency of occurrence in text. A number of more sophisticated variations evolved from this original theme such as combining location with occurrence, requiring co-occurrence within pre-determined locations, associating frequency of occurrence with subject categories, and so on.

In this project frequency of occurrence was used in the computation of weights. The computer performs the simple computation of calculating number of occurrences per thousand text words and then converts the resulting fraction into an index number (weight) in the following way:

If the frequency of the term per thousand words of article is less than or equal to 1, the article is assigned a weight of 1. If the frequency of the term per thousand words is greater than 1 and less than 3, then it is assigned a weight of 2. Finally, if the frequency of the term per thousand words is greater than or equal to 3, then it is assigned a weight of 3.

Weights	Frequency per thousand words
1	$\leqq 1$
2	> 1 and < 3
3	$\geqq 3$

Frequency of occurrence in case of a drug means the total number of occurrences under any of its names and in case of a non-drug term frequency of occurrence under any of its synonyms. Whether or not this is a meaningful and useful method for the assignment of weights to index terms to indicate their relative importance in the subject description of the document remains to be determined through the evaluation of the output.

Forms of Output

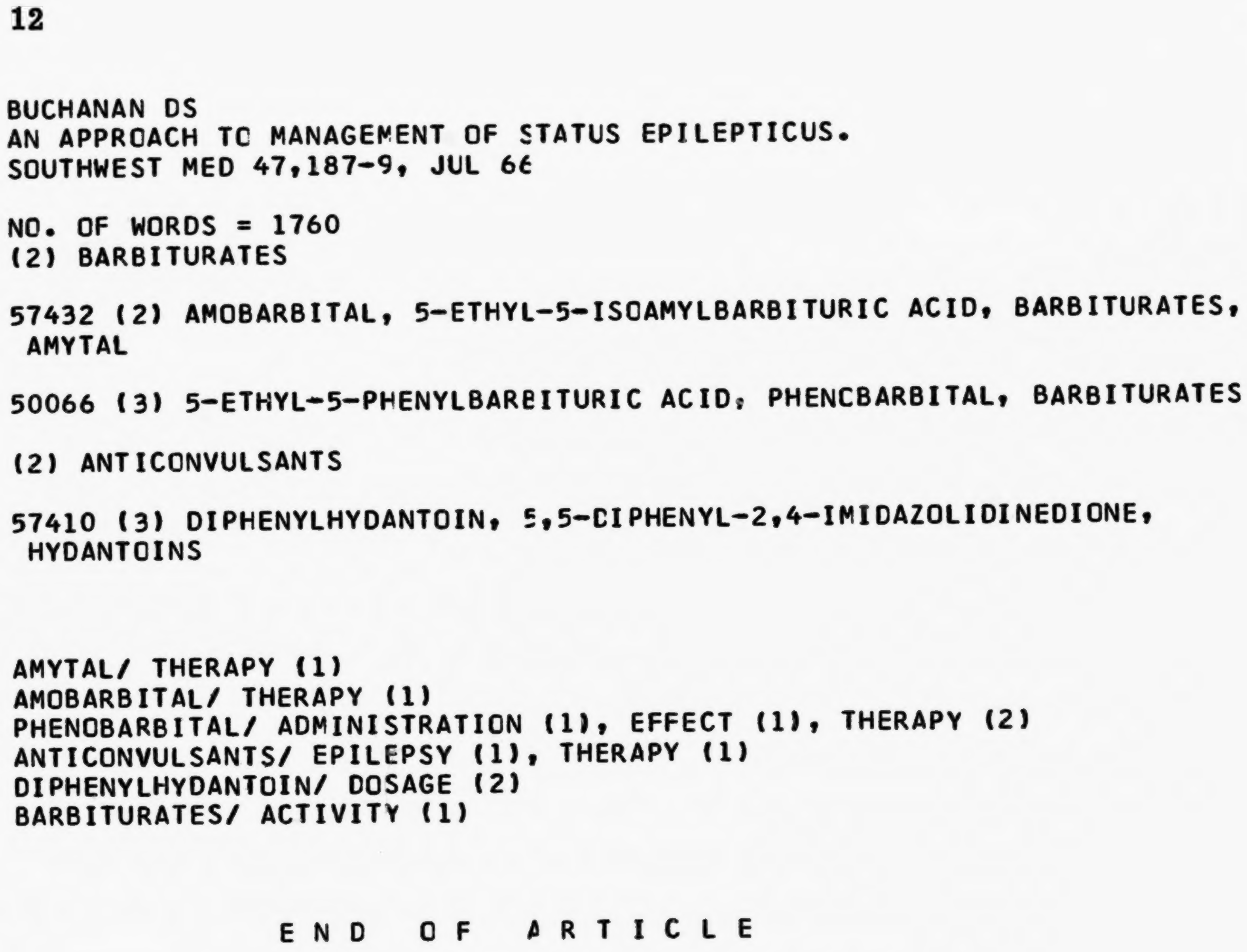
The indexing program was designed with two kinds of output in mind, one based on the other. In other words, the machine file had to be designed to be satisfactory for search purposes and at the same time produce a printout with a format that is both pleasing to the eye and convenient to use. A decision was made to put the index records on magnetic tape which in this case will mean that the entire collection of records is searched each time a query is run, with each record being examined for each query.

For each document the printed record contains the following elements, in this order:

author
subject
bibliographic citation
total number of text words
index terms without modifiers, with their respective weights and Chemical Abstracts registry numbers
index terms with modifiers, with their respective weights

The fact that each record shows all index terms which were assigned to the document is useful because the terms taken together add up to a rudimentary abstract. Certainly these are no substitutes for good informative abstracts prepared by good human abstractors, but they do have a certain degree of usefulness.

A description of the magnetic tape index file is given later in this report. A detailed discussion of search capabilities will be given in the next progress report after completion of the design and implementation of the search program.



```
12

BUCHANAN DS
AN APPROACH TO MANAGEMENT OF STATUS EPILEPTICUS.
SOUTHWEST MED 47,187-9, JUL 66

NO. OF WORDS = 1760
(2) BARBITURATES

57432 (2) AMOBARBITAL, 5-ETHYL-5-ISOAMYLBARBITURIC ACID, BARBITURATES,
 AMYTAL

50066 (3) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

(2) ANTICONVULSANTS

57410 (3) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE,
 HYDANTOINS


AMYTAL/ THERAPY (1)
AMOBARBITAL/ THERAPY (1)
PHENOBARBITAL/ ADMINISTRATION (1), EFFECT (1), THERAPY (2)
ANTICONVULSANTS/ EPILEPSY (1), THERAPY (1)
DIPHENYLHYDANTOIN/ DOSAGE (2)
BARBITURATES/ ACTIVITY (1)

            END  OF  ARTICLE
```

FIG. 1

IV. DEVELOPMENT OF THE INDEXING PROGRAM

Hardware Considerations

In developing a system of the scope and complexity that results from the synthesis of many sub-programs into an operating automatic indexing program one is inevitably concerned with the characteristics of the computer on which the system of programs is to run.

Three classes of computer hardware characteristics are of particular relevance to the system designer. The first relates to the speed and capacity of the central processor, the second concerns the structure of available storage and the third is the character or word organization of the computer's main memory.

The choice of computer for this project was dictated by the availability of an IBM 7040 computer which is part of the installation of the Center for Computer and Information Services (CCIS) at Rutgers University.

Since this project is an experiment of limited size the matters of computer speed and storage are not of prime concern. However, the concept motivating this approach has been that of a pilot project for which the programmed solution should be relatively easily applied to a large scale production situation. The IBM 7040 Data Processing System has a basic cycle time of 8 microseconds. The size of the main computer storage is 32K words.

Test runs of the automatic indexing program indicated that the average time to process one thousand words was 18 seconds. Computers are now available which would increase the processing rate by a factor of at least ten. Thus a third generation computer with an 800 nanosecond access time would be able to index a hundred medical articles containing 225,000 words in about six or seven minutes.

Since the size of computer high speed main storage is relatively limited the amount of information accessible at high speed is limited. In the IBM 7040 32,768 words each of thirty-six bits are addressable and randomly accessible. However in practice the storage available is reduced to about 20,000 computer words by the necessity to store the IBSYS operating system software. Because the thirty-six bit computer word can accommodate 6 characters it can be taken as an average text word and we have room in memory to store about quarter of an average size novel. Of necessity some room is taken up by the text processing program. The first important question is to decide where the dictionary of medical drug terms and related words is to be stored and what form of file is to be used to maintain high speed processing and yet not sacrifice storage. In a sense the simultaneous conservation of space and maintenance of high speed are incompatible objectives in programming. The more tightly variably sized information is packed the more time will be sacrificed in unpacking it for processing.

The problem of capacity is complicated by the existence of a variety of types of storage media ranging from magnetic ferrite cores in the central processor through the peripheral storage devices of disk and drum to magnetic tape. Neither punched cards nor paper tape are considered as storage in this project. Punched cards in this project were only used for low volume input, for example, updating the dictionary tape. Paper tape input could be used for the same purpose.

The CCIS computer has an IBM 1301 Random Access Disk Unit consisting of 2 modules having a capacity of 36 million words, a transfer rate of 90 ~~75~~Kc and a maximum access time of 218 ~~750~~ milliseconds. The system has no drum storage but has seven 729-Mod 5 magnetic tape drums with transfer rates of 60,000 cps and tape density of 800 bits per inch.

Types of storage and their capacity vary from machine to machine and installation to installation. It is possible to design a very efficient system for a particular computer that would become inefficient if transferred to a similar computer having less storage capacity. For instance, it is not envisaged that the program would be run on a smaller computer such as the IBM 1401's which are satellites to the center's 7040. But it is considered that it might well be transferred to a faster computer with greater storage capacity.

An important design criterion in automatic indexing is to minimize cumulative access time, that is, the time required for the processor to retrieve and store all text words and dictionary words with which it must work in processing an article or a series of articles during a particular run. This necessitates the packing of information into high-speed storage to make the best possible use of this valuable asset. In the present project it was possible to store the whole dictionary in main memory and to program a very rapid access. The basic concept provides for practical growth of the project. If the dictionary were to become too large for memory it could be placed on the disk unit. The program is written in such a way that only minor modifications would be necessary to adapt it to incorporate disk storage. On the other hand capacity constraints may soon become insignificant with the advent of large-capacity fast-access storage.

Sequential processing versus random access. To solve the capacity problem the system designer must weigh the pros and cons of the available storage devices. Four media are considered in this context, namely, high speed core, random access disk storage and two sequential storage media, high speed magnetic drum and magnetic tape.

All data is equally accessible in main memory at the same highest speed. Hence the designer is only concerned with size constraints. If memory is too small auxiliary storage is required. Searching a magnetic tape in the random fashion required to match a term in text with the identifying term in the dictionary would be prohibitively slow. Sequential searching of records on magnetic tape is feasible in the second project phase of searching the file of index records and becomes more attractive when all queries are being processed simultaneously.

Disk has the advantage that all the information is randomly accessible but since it involves a mechanism with considerable movement of heavy parts access is a couple of magnitudes slower than core and moreover the access time depends on the position of the information. When the capacity of core is exceeded it would be possible to put the dictionary on a high speed drum, sequential storage medium which can be randomly accessed. This possibility is not really considered in this case even though a high speed drum can be faster than disk. Availability could be the deciding factor and disk is available in the CCIS installation.

Computer manipulation of information. It is impossible to ignore the organization of the computer memory and the repertoire of instructions. One obvious distinction is the word or character orientation of memory.

If information is handled in words as in the IBM 7040 it is necessary to be able to pack and unpack words and to shift characters within the words. The programming will be facilitated if there are character handling instructions, shifts and logical instructions. If the computer is character addressable the packing and unpacking of words is eliminated and character manipulation is simplified. The tendency in third generation machines is towards flexible handling of information in a variety of variable sized modules or bytes.

The effective operation of an automatic indexing program is greatly facilitated by using techniques that take advantage of economical data packing and efficient logical manipulation of information.

Software Considerations

The choice of computer language is one of the most important decisions to be made by the designer of an automatic indexing system. This section discusses briefly the relative merits of machine oriented languages, procedure oriented languages and problem oriented languages. Text processing, automatic indexing and index file searching are more closely related to data processing than to 'scientific' programming involving numerical methods. Little computation other than simple counting is required. The most frequently used functions are reading and writing of file records, sorting of information using a particular collating sequence, character handling, word composition and decomposition, logical decision making and list processing.

Although the following discussion refers to the available IBM 7040 the arguments used are general in nature and could be applied to many computers. To the basic concept of flexibility and ability to expand into practical volume production must be added another important element of the philosophy of the project--to be as independent as possible of the particular computer and to achieve as high a degree of generality and compatibility as possible in the program.

Machine oriented languages. The use of absolute machine coding or an assembly language such as MAP -- Macro Assembly Programming enables a competent system designer to take full advantage of the capability and capacity of the computer he is using. Certainly specialized instructions may be available which significantly save execution time, or facilitate character manipulation. Bits, bytes or characters may be efficiently handled, core storage can be conserved and input-output time may be pared down.

It is hard to believe that anyone currently writes in machine code yet over-enthusiastic programmers in the not too distant past have strenuously claimed to write better code than any 'automatic' method could possibly produce. The credibility of such claims may be heavily discounted in all but a minute number of circumstances. The tedious representation of long strings of binary bits by means of octal or decimal code is extinct as a method of coding computer programs. The setting of bits to indicate various options obscures any mnemonic quality and is highly subject to error in numeric coding.

An assembly language like MAP is largely a one-for-one mnemonic representation of machine code instructions; in addition there are macro instructions. Programming is faster and many of the clerical type errors made in machine code are eliminated by the assembly process. There are some major disadvantages in the use of assembly language. It would be inconsistent for example, to devise a system of automatic indexing and file search dedicated to efficient documentation and information processing by using a program that would be unintelligible to a majority of intelligent professional people. This is a real drawback to the use of assembly language. Only the computer programmer has any idea what the program is trying to do. This creates a comprehension gap between the professional devising the system and the programmer which the latter is frequently reluctant to bridge.

Another distinct disadvantage is that the program is not transferable to a different kind of computer. Moreover the existing program would prove of little help in the major rewrite job entailed in producing a program for the new computer.

It may be argued that those who design the system do not need to understand the program. If the view is taken that it is highly desirable for the designer of the system to be convinced that his stated concepts are being understood and implemented by the programmer review of progress will be necessary and assembly language is virtually ruled out of consideration.

Procedure oriented languages. The most evident advantages of procedure oriented languages are that they are independent of the computer and more closely resemble the everyday language of the user. A generalized compiler language such as FORTRAN or COBOL can be translated into different machine languages using appropriate processors.

Compiler languages share some of the advantages of assembly language; namely, reduction of clerical errors, use of proven functions and subroutines and error detection. They also have significant advantages over assembly language. They are:

1) Vastly improved documentation if properly used.
2) Simplified communication with the computer.
3) Emphasis placed on problem rather than computer.
4) Significant saving of program preparation time.
5) Time savings in program error correction or debugging.
6) Reduced machine time spent in program testing.
7) High degree of compatibility facilitating easy transfer of program to other computer systems.

The drawbacks of compilers involve inefficiency in the use of core storage, constraints imposed by the conventions followed in using the compiler and lack of access to some of the more ingenious machine functions.

The need for logical instructions for use in bit manipulation not provided in FORTRAN IV is recognized by the provision of three machine-dependent logical instructions AND, OR and NOT for the IBM 7040 FORTRAN. Thus it is possible to write all the bit manipulation routines necessary for character handling. In addition it is possible to write and call MAP subroutines to perform specialized manipulation.

COBOL is also available with the 7040 IBSYS and provides excellent documentation and powerful editing features in the data division. But COBOL takes longer to write and more time to debug and considerably longer to compile than the equivalent program written in FORTRAN.

Problem oriented languages. In the hierarchy of computer languages problem oriented languages stand above the compiler languages. There are many such languages devoted to the solution of problems of certain types such as simulation, civil engineering or coordinate geometry. Since these languages cater to particular professional groups they may be said to be user-oriented.

Both SNOBOL and COMIT are user-oriented general purpose symbol-manipulation programming languages. These languages are particularly convenient for an ever widening group of non-numerical problems ranging from text-processing to information retrieval, and from theorem proving and pattern recognition to the maintenance of largely non-numerical files.

The use of procedure oriented languages minimizes the user's time for programming a problem since the symbol-manipulation languages are well suited for text-processing or automatic indexing. On the other hand there are some definite disadvantages. One drawback is the extremely long compilation time because they are general purpose languages not specifically tailored to the job of automatic indexing, and another is that processors for these languages have been implemented for only a restricted set of computers.

Languages used in this study. On the basis of the foregoing arguments the main programs were written in FORTRAN. Arrays and matrices are particularly easy to deal with in FORTRAN and there are a number of powerful functions available so that just about any effect can be easily achieved. Originally the idea was to apply MAP assembler language selectively in such a way as to achieve almost optimal use of the machine capabilities.

Very often the advantage of writing a routine in MAP did not appear to outweigh the trouble of working with two languages. The introduction of machine oriented routines reduces the compatibility of the program and necessitates rewriting when transferring to another computer.

Any advantage derived from speeding up running time is offset by the other disadvantages. Examination of the MAP listing of the compiled FORTRAN program did not suggest any obviously great economies. However, if an exceptionally efficient MAP program were desired work on the compiled listing might be the best approach.

The Overall Programming Approach

As it was described in the section on the dictionary terms included in the dictionary belong to two broad categories -- those which are names of drugs and those which are other terms than the names of drugs.

An interesting characteristic of the dictionary that should be noted here is the extreme variation in the length of drug names from four letters as in PHOB to sixty-five alpha-numeric characters as in P-(TETRHYDRO-2H-1, 2-THIAZIN-2-YL)-BENZENESULFONAMIDE, S, DIOXIDE.

This condition brings into focus the question of how to accomodate the dictionary in memory and how to access the terms for comparison with text words. Two main considerations thus emerge at this juncture; the economical use of space in storing the dictionary and of time in processing the text. In general memory storage is saved at the expense of processing time. A simple example of this is the use in this project of an integer to represent a transfer address which is known to be less than or equal to 4096 or 2^{12}. It would be possible to represent any of these integers with 12 bits instead of a whole computer word of 36 bits. Consequently it would also be possible to pack three numbers into a word and then extract each number and place it right justified in another word before it could be used for its purpose as an integer.

The observation may be made that extraordinary efforts to save memory are futile when there is plenty; it pays rather to take advantage of the space available and strive for fast processing.

Allotment of a constant sized field for each dictionary term would simplify programming the search for matching terms. Since these fields would have to be 12 computer words to accommodate the largest terms there would be a considerable waste of storage. Moreover, this unused storage would be inaccessible. This would very much restrict the size of dictionary that memory could hold.

The alternative is to conserve space by placing one term immediately after another. Clearly, because of the variation in size of words, separators, links or pointers will be needed to find the beginning of the next word in memory.

The concept of the table look-up is a very powerful one in data processing and proves very much faster than decision logic for searching a table or file when the query is random. In brief, a word in text has to be matched by an identical one in the dictionary. By using the first two letters of the word a numeric index can be derived which is the value of a transfer address which contains the address of the first term in the dictionary beginning with the given letters. These transfer addresses are stored in a vector or matrix. This method leads immediately to the locality where the word may be found if it is in the dictionary. Full details of this fast combined look-up and search are given in the section on dictionary processing.

When a word in the text is matched with a drug term in the dictionary it will be recorded in the index record by means of a package of terms which consists of the registry number, chemical name, generic name and group name. If the term found is a trade name it must be recorded as well. Of course, this package, to be referred to as type 1, is recorded only once in the index record but a frequency index is calculated from its number of occurrences per thousand words of text.

If the appropriate package of terms were recorded in the dictionary with each drug term the size of the dictionary would be about quadrupled. The solution to this problem of redundancy is to record a pointer to the associated type 1 package with each drug term in the dictionary. In addition each package is recorded only once instead of an average of eight times. A further saving of storage is achieved through recording dictionary addresses of the package terms instead of the actual terms. Moreover the uniformly sized addresses each occupying a computer word simplify programming.

Consequently each term in the dictionary is preceded by a linkage to the next term and a pointer to its associated package. The type 1 packages are recorded after the dictionary and are followed by the type 2 associated terms.

The dictionary term is keypunched into a card together with the address of its package. The type 1 and type 2 package addresses were allocated by the programmer assigning numbers to the packages. These numbers would be converted to index numbers by the program. However, the actual process could be automated as described in the section on dictionary processing.

Dictionary processing first requires the cards to be written to magnetic tape and then sorted according to the IBM 7040 scientific collating sequence using the system sort and merge IBSORT package. The tape is read by the dictionary processing program to create a matrix of transfer addresses from the first two characters of each term, place the term in memory, calculate linkages, put the package pointer in place and read in the vectors for the type 1 and type 2 packages. When all of this processing is completed all these things are written out to tape creating a binary record which is then ready to be read into memory by the automatic text indexing program.

The Automatic Text Indexing Program

A detailed flow-chart of the automatic text indexing program appears in Fig. 2.

The program first reads the binary tape of the current version of the processed dictionary into memory. The medical drug articles have been keypunched into cards with the text occupying columns 1 to 72. The heading of the article consists of author, title and citation followed by a sentinel card which separates the heading from the body of the article.

A parameter card can be read in to specify the length of a line in the printed record of the article heading and another parameter specifies the length of a line in the body of the index record. No line justification is attempted in either case, but no word is hyphenated by the program.

The program reads in a card containing 72 characters of text at a time and processes the information from that card. If the volume of articles to be indexed became considerably larger it would be worthwhile to perform a card-to-tape conversion and then the indexing program could read the records from magnetic tape at a much faster rate.

A word from the card record is placed in a buffer character by character and the number of characters counted. Since the smallest word occurring in the dictionary contains four letters any text word containing three letters or less is discarded immediately.

A subroutine is called upon to compose computer words of six characters each from the characters of the textword being examined.

In the IBM 7040 words have to be compared since characters do not exist on their own as addressable entities.

The dictionary terms are also stored in computer words so that the composition of the characters of the textword into computer words enables time spent in comparing words to be reduced by a factor of six.

An index is generated from the first two characters of the text word which points to

FLOW CHART OF AUTOMATIC INDEXING PROGRAM

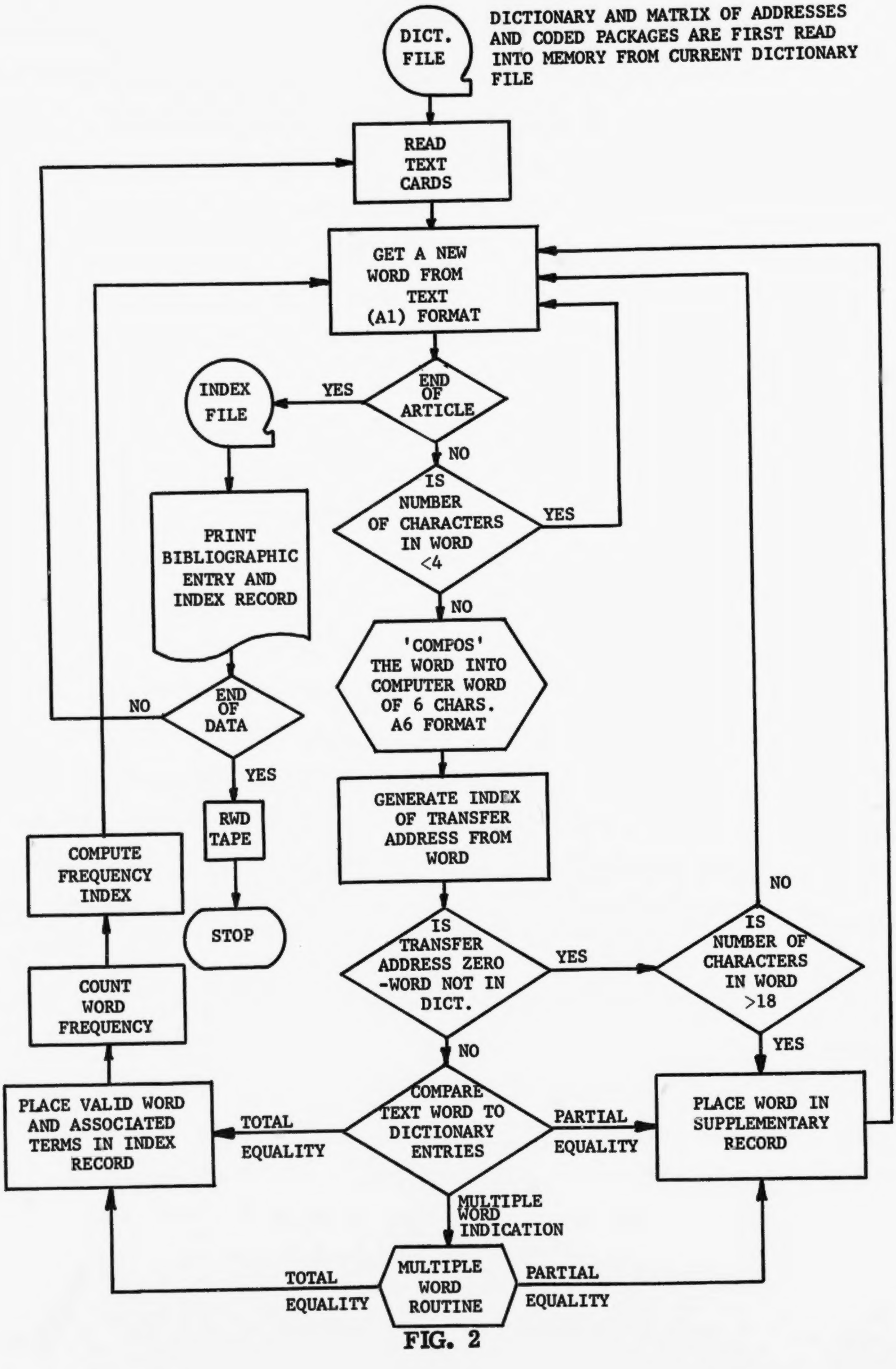

DICT. FILE
DICTIONARY AND MATRIX OF ADDRESSES AND CODED PACKAGES ARE FIRST READ INTO MEMORY FROM CURRENT DICTIONARY FILE
READ TEXT CARDS
GET A NEW WORD FROM TEXT (A1) FORMAT
END OF ARTICLE
YES
NO
INDEX FILE
PRINT BIBLIOGRAPHIC ENTRY AND INDEX RECORD
END OF DATA
NO
YES
RWD TAPE
STOP
IS NUMBER OF CHARACTERS IN WORD <4
YES
NO
'COMPOS' THE WORD INTO COMPUTER WORD OF 6 CHARS. A6 FORMAT
GENERATE INDEX OF TRANSFER ADDRESS FROM WORD
IS TRANSFER ADDRESS ZERO -WORD NOT IN DICT.
YES
NO
IS NUMBER OF CHARACTERS IN WORD >18
NO
YES
COMPARE TEXT WORD TO DICTIONARY ENTRIES
TOTAL EQUALITY
PARTIAL EQUALITY
MULTIPLE WORD INDICATION
PLACE VALID WORD AND ASSOCIATED TERMS IN INDEX RECORD
PLACE WORD IN SUPPLEMENTARY RECORD
COUNT WORD FREQUENCY
COMPUTE FREQUENCY INDEX
MULTIPLE WORD ROUTINE
TOTAL EQUALITY
PARTIAL EQUALITY

FIG. 2

the matrix of transfer addresses. A zero in the transfer address indicates that the word is not in the dictionary. A high proportion of words is immediately discarded in this way. However if this word contains more than 18 characters it is classed as a "long word" and there is an option in the program that provides for a printed listing and punched card output containing the long word in columns 1-72 and the serial number of the article in the file in columns 76-80.

This output is available for human examination by the information retrieval researcher who can select the terms he wishes to add to the dictionary and use the retained cards to update both the dictionary and the index file record.

If the transfer address is not zero it is then the address of the first dictionary word having the same first two letters as the text word. The next step is to compare the text word with the dictionary entries. Since these terms are being compared computer word by computer word it is possible to record partial equality of terms and the degree of identity. Because these partial comparisons could produce terms that should be considered for inclusion in the dictionary an option is available to write out and punch out a supplementary record for subsequent examination.

Some of the dictionary terms contain more than one word, for example, 5-ETHYL-5-PHENYLBARBITURIC ACID. If the text word gives apparent equality with the dictionary term the program provides a multiple word routine which will examine the next text word for equality with the next word in the dictionary entry.

When a valid word is found the word and its associated package is placed in the index record.

To record the relative frequency of occurrence of drug names in the text of an article a valid word is counted each time it occurs. The final absolute count is used in the calculation of weights as it was discussed earlier.

When the end of an article is read the index record is printed out and also written to magnetic tape. The input format for the text indexing program is shown in Fig. 3.

Data and character handling subroutines. The initial actual programming approach was to write a number of subroutines that would perform necessary data and character handling functions. This approach facilitated program testing in the early stages of the project. Brief descriptions of the subroutines are given below and program listings are included in Appendix 8.

If characters are read into the IBM 7040 in A-1 format then each character is placed in

TEXT EDITING-INPUT FORMAT

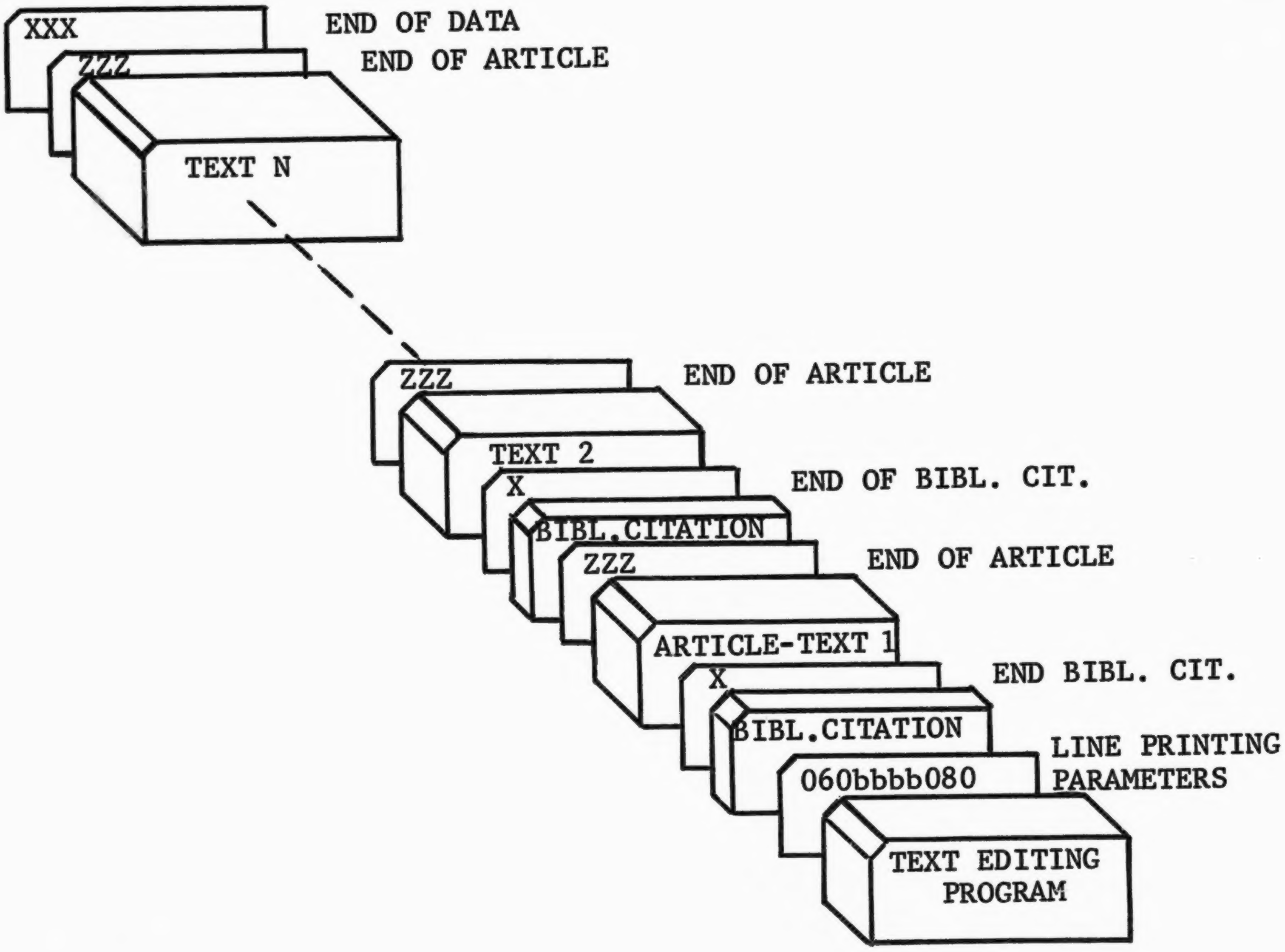

XXX
END OF DATA
ZZZ
END OF ARTICLE
TEXT N
ZZZ
END OF ARTICLE
TEXT 2
X
END OF BIBL. CIT.
BIBL.CITATION
ZZZ
END OF ARTICLE
ARTICLE-TEXT 1
X
END BIBL. CIT.
BIBL.CITATION
060bbbb080
LINE PRINTING PARAMETERS
TEXT EDITING PROGRAM

FIG. 3

the left most character position of a six character computer word in main storage. The other five character positions will contain spaces. Thus DILANTIN would appear as

D I L A N T I N

Clearly such a representation is wasteful of space. However, comparisons performed character by character are wasteful of processing time.

COMPOS

This subroutine takes any given number of characters recorded singly and packs them into the required number of computer words.

DECOMP

This subroutine performs the inverse function of decomposing or unpacking a given number of computer words into the required number of characters recorded singly.

INDEXT

This subroutine constructs a numeric index, 1 through 4096, from the first two characters of a word.

HEAD

This subroutine processes the heading of an article for printed output. It will read a parameter specifying the maximum number of characters per line. Authors, bitte and citation are printed on separate lines and no word or term is broken or hyphenated by the subroutine. A program to perform hyphenation would prove too complicated at this stage. Lines of more uniform length could be obtained by adjusting the spacing between words so that they will occupy the maximum number of characters permitted.

CONCOD

This subroutine converts the binary code used for economical representation of the drug register numbers into BCD, binary coded decimal form required for printing these numbers on the same line as other alphanumeric BCD information.

CARD-TO-TAPE

This subroutine transcribes the drug terms keypunched into cards on to magnetic tape for sorting and merging with the dictionary tape before the dictionary can be updated and processed.

WRDPUT

This subroutine checks whether a valid word of given length can be fitted into the print line buffer and calls the subroutine DECOMP to convert the word from A6 format to A1. If there is no room for the word in the print line, the line is printed out in the index record while the line buffer is reinitialized for the next line to be printed and the valid word becomes the first word in it.

Rules for keypunching. If the aim of this project is to achieve as high a degree of automation as possible it is necessary to look forward to a time when a computer readable record of an article is a by-product of the process of printing it. In this case it would be necessary to write programs to handle such a record. In compiling rules for keypunching articles the wide discrepancy between the variety of characters available in the printed record and much smaller number in the character set of the model 26 card punch. It was decided not to introduce the complications of using special codes to represent the particular printing characters and symbols not available on the punch for this would have added considerably to the cost and difficulty of keypunching.

The idea was to use as few rules as possible but then to make it as true to the original as the rules would allow. An example of an incompatibility that necessitated a rule is the fact that when a word ends on the right hand side of a printed line the next word in the text starts at the beginning of the next line. In the punched card this means one word ending in column 72 and the next starting in column 1 of the next card. Since these are treated as successive columns the two words would be treated as one long word. Rules for keypunching articles are listed in Appendix 10.

Author card. The author's surname is punched first, followed by a space and the author's initials in adjacent columns. If there are several authors a comma and a space separate the names.

Title card. The full title of the article is punched into the title card. If more than 72 columns are required the rest of the title is punched into following cards.

Source card. Into this card is punched the following source information: publisher name and location, title, volume, pages, date of publication, etc. according to usual bibliographic practice.

The bibliographic entry read in on cards before the text of an article can be printed out by means of the subroutine HEAD which was described earlier. There is an option available for varying the length of the printed line of the bibliographic entry by specifying a line length parameter. A card containing an X in the first column signals the end of the bibliographic entry. Fig. 4 shows the punched card record for the bibliographic entry.

Dictionary Processing and Updating

Placing the dictionary in storage. The dictionary term is keypunched, left justified into the first 72 columns of a card and the 'package' code right justified into the last five columns. The 'package' code is a positive or negative integer used in processing the writing of associated terms to produce index records from dictionary terms found in the medical article being processed.

Example of card

columns 1 through 72	76 - 80
3-ETHYL-5-PHENYLHYDANTOIN	19

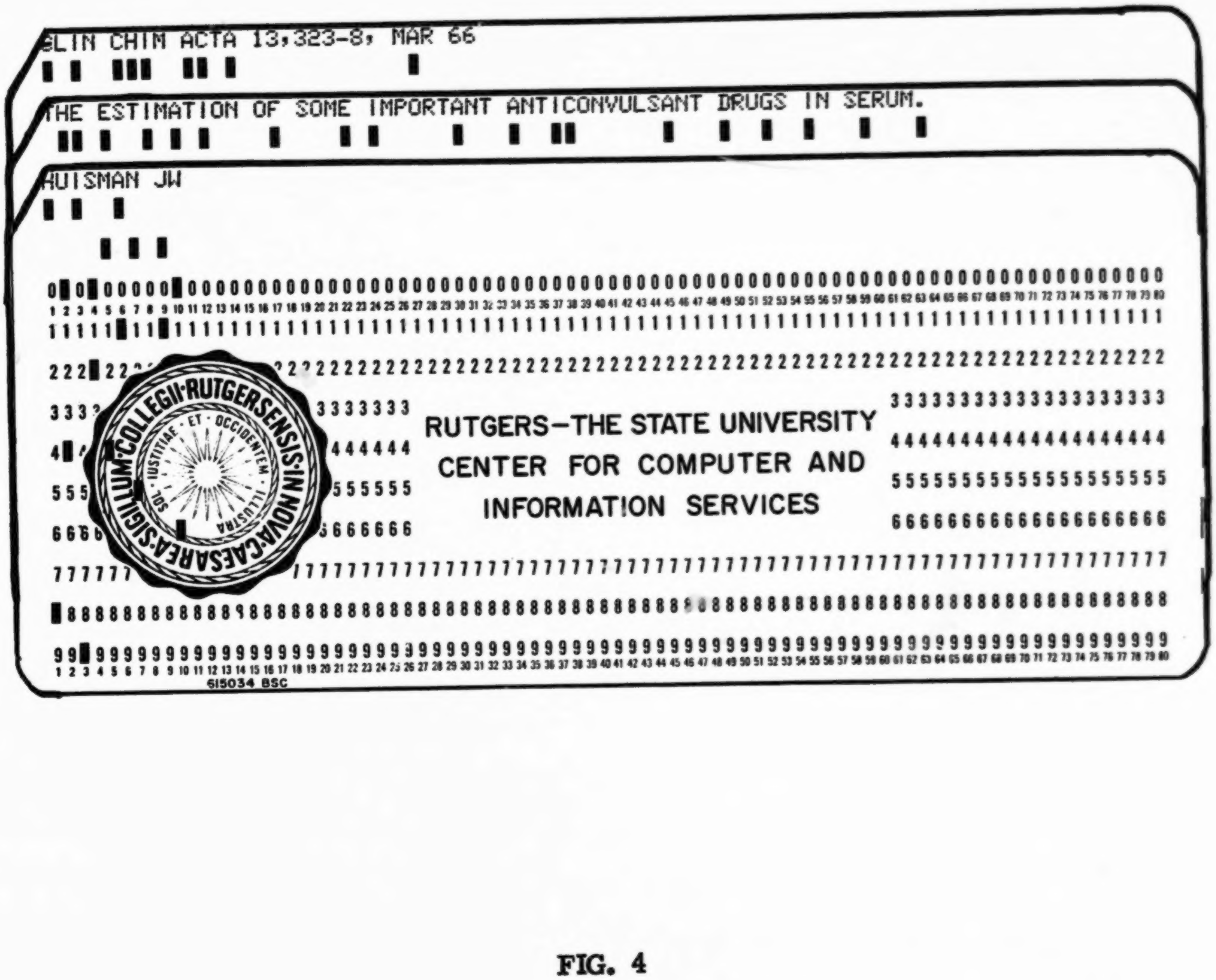
The first sub program in the sequence of dictionary processing programs is the card-to-tape conversion which transcribes the dictionary cards to magnetic tape.

Collating sequence--IBSORT--Generalized Sort and Merge. When the previous program reads the sentinel card containing the word SIGNAL control passes to the generalized sort and merge program, IBSORT. This program allows the use of one of two collating sequences, the so called 'scientific' sequence and the commercial. The scientific collation was selected and sorting was performed on the maximum possible field of sixty characters. Caution is necessary, however, in treating computer words with a letter in the first six bits as integer numbers as was required later in the text processing program. The normal sequence of letters are in ascending order of magnitude with B>A and C>B until J is reached. Any word beginning with J or any following letters treated as an integer is read as a negative number because it has a bit in the most significant position. Hence, in any routine having to check the correct alphabetic sequence it is necessary to take account of this transition.

There are two circumstances involved in this phase; the creation of a sorted dictionary tape and the updating of the dictionary tape by the addition of new terms. In the former case the magnetic tape produced by transcription of the dictionary cards is sorted. In the latter case the tape containing new entries is sorted then merged with the old dictionary tape to produce a new updated dictionary tape.

Dictionary access concepts--matrix of transfer addresses--linkages. The important

PUNCHED CARD RECORD FOR BIBLIOGRAPHIC ENTRY



CLIN CHIM ACTA 13,323-8, MAR 66

THE ESTIMATION OF SOME IMPORTANT ANTICONVULSANT DRUGS IN SERUM.

HUISMAN JW

RUTGERS—THE STATE UNIVERSITY
CENTER FOR COMPUTER AND
INFORMATION SERVICES
615034 BSC

FIG. 4

functions to be performed in the processing of the sorted dictionary tape are the following.

1. To create a matrix, actually, for simplicity a vector or array of transfer addresses. In the FORTRAN text processing program these addresses will be used as subscripts to locate terms in the dictionary array.
2. To create pointers or linkage to successor terms in the dictionary.
3. To record the associated 'package' number for each term.
4. To read in the successive dictionary terms into contiguous portions of the dictionary array.
5. To read the 'packages' into arrays recording the dictionary address of a term instead of the term itself.

A detailed flow chart of the dictionary processing is shown in Fig. 6.

A step by step description of the processing follows. The first term to be read from the sorted dictionary is

1-AMINOPROPANE-1,3-DICARBOXYLIC ACID

This term takes up 36 character locations or 6 words. The two first characters of this term 1- generate an index of 0110 octal or 96. One is added so that the possible indices that could be generated range from 1 to 4096 thus avoiding the remote possibility of a zero subscript. Consequently the initial pointer or linkage in the matrix at the location 97 is set to 3. The package number associated with this term is -5. When the number of computer words in the term is counted and account taken of the fact that two computer words separate the dictionary drug terms then the pointer or index to the first computer word of the successor term may be computed and in this case it is 2 + 6 + 2 + 1 = 11. Thus the first term will appear in memory as in Fig. 5 below.

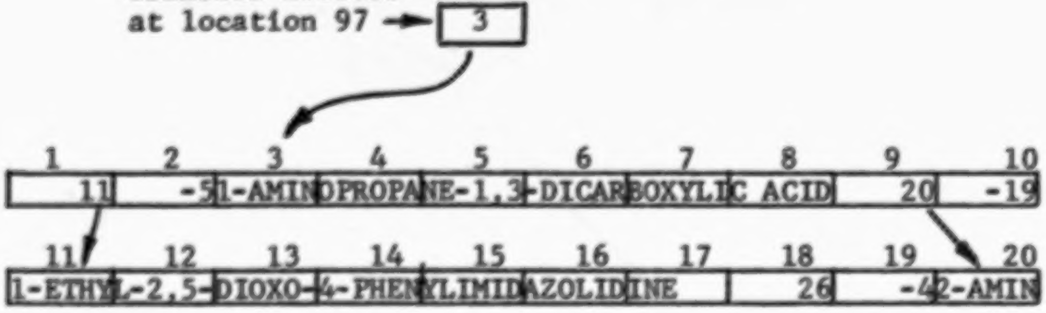


FIG. 5

The dictionary array is a linked set affording easy access to the next term from the preceding term.

When all the dictionary terms have been arranged in memory they are written out to the processed dictionary tape together with the number of computer words occupied by the dictionary array.

The dictionary packages have to be processed into arrays in the following manner. For example, the 5th package of type 1 contains the following cards:

Card 1	5 56860
Card 2	GLUTAMIC ACID
Card 3	1-AMINOPROPANE-1,3-DICARBOXYLIC ACID
Card 4	BLANK CARD

The information in this package is recorded in the type 1 array as 56860 996 3 0

It will be observed that the location 3 of the chemical name itself is recorded here. The minus sign in front of the package number in the second word of the dictionary indicates that if this term is located in the text of an article it does not have to be written out in addition to the package. In this case it is recorded in the package and naturally, does not have to be recorded twice. Fig. 5 shows that the second term begins in location 11 while the third term begins in location 20 and the fourth in location 26. The completely sorted dictionary of drug terms together with their associated address location and package numbers are listed in Appendix 4.

As has been previously stated one of the concepts underlying the program was to think in terms of expandability to a large scale production operation without having to make drastic alterations in the program logic. In a large scale production operation the dictionary might have grown too large to be accommodated in core storage. However, the matrix of transfer addresses could certainly always be stored in main core. An enlarged dictionary would be written out to auxiliary mass storage, probably a disk unit. The vector of addresses would now provide appropriate pointers to the required sectors.

If this kind of expansion had to take place in the immediate future the IBM 1301 Disk Storage on the IBM 7040 would have to be used. The organization of data in disk storage and the method of processing data affect the seek time for a given operation. The disk storage is organized in cylinders and the access mechanism requires time to move from one cylinder to

DICTIONARY PROCESSING

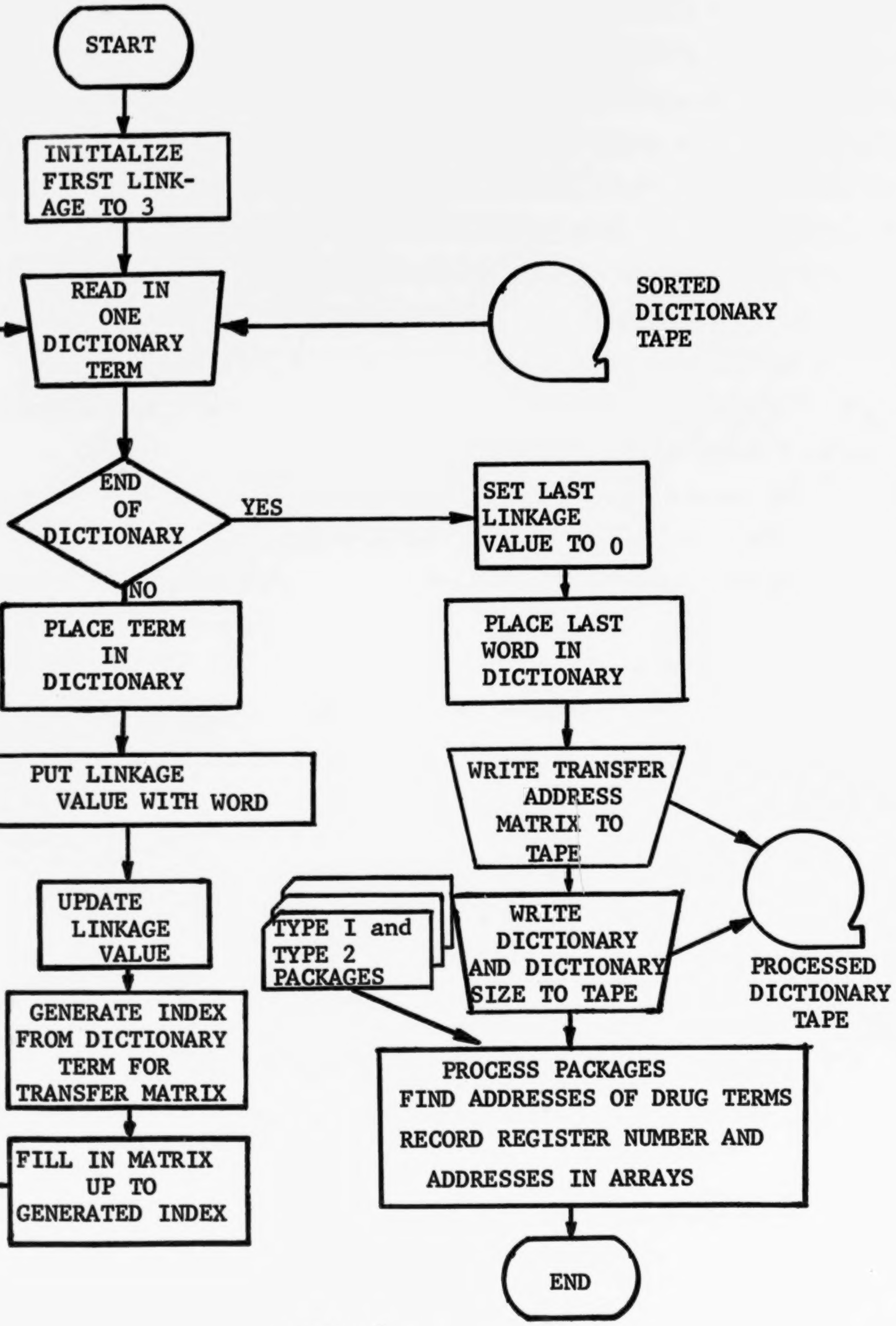

START
INITIALIZE FIRST LINKAGE TO 3
READ IN ONE DICTIONARY TERM
SORTED DICTIONARY TAPE
END OF DICTIONARY
YES
NO
SET LAST LINKAGE VALUE TO 0
PLACE TERM IN DICTIONARY
PLACE LAST WORD IN DICTIONARY
PUT LINKAGE VALUE WITH WORD
WRITE TRANSFER ADDRESS MATRIX TO TAPE
UPDATE LINKAGE VALUE
TYPE I and TYPE 2 PACKAGES
WRITE DICTIONARY AND DICTIONARY SIZE TO TAPE
PROCESSED DICTIONARY TAPE
GENERATE INDEX FROM DICTIONARY TERM FOR TRANSFER MATRIX
PROCESS PACKAGES
FIND ADDRESSES OF DRUG TERMS
RECORD REGISTER NUMBER AND ADDRESSES IN ARRAYS
FILL IN MATRIX UP TO GENERATED INDEX
END

FIG. 6

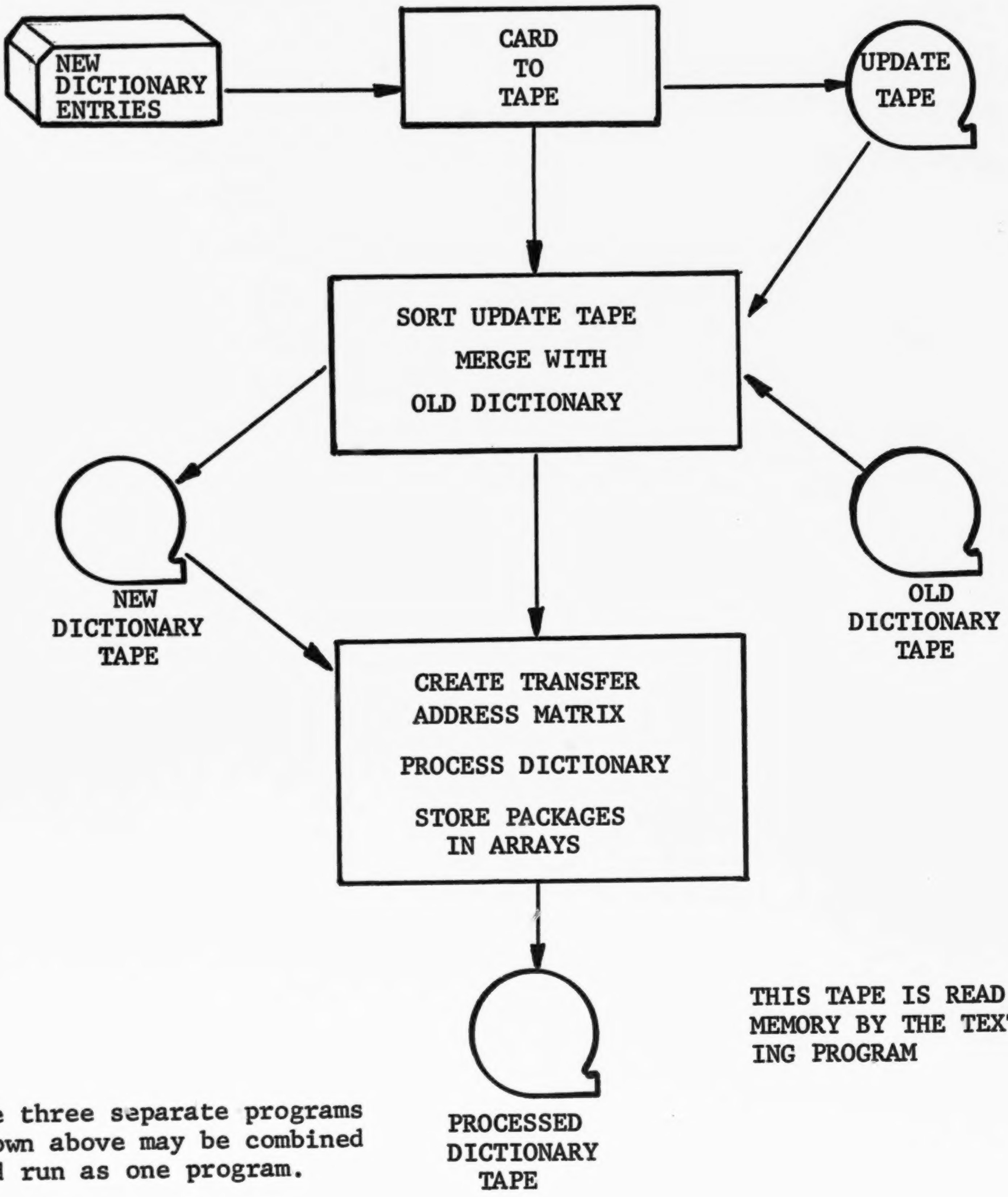
another. A disk module containing 250 cylinders is organized into 5 areas and these are divided into six sections. Access motion time is 50 milliseconds within a section of one area while from one area to another requires 180 milliseconds.

These access times are several thousand times greater than the 8 microseconds required to access main storage. However, a very much enlarged dictionary could be accommodated on disk and the program could be very easily adapted to this change.

Dictionary maintenance. When it becomes necessary to add new terms to the dictionary these should be keypunched into dictionary cards in the format already described. New packages of drug synonyms and new package members would have to be allocated. The dictionary cards would be then transcribed to tape and the dictionary processing program called into play using the sorted tape and the packages cards as input.

An option in the text editing program provides the basis for a man/machine method for dictionary up-dating. Fig. 7 describes the process of dictionary up-dating. The program permits the punching of "long words" into cards as computer output. The cards may be interpreted or listed so that they may be examined for possible additions to the dictionary. When the suitable additions are selected the cards have to be coded with package numbers and treated in exactly the same manner as described for new cards in the section above.

DICTIONARY UPDATING


NEW
DICTIONARY
ENTRIES
CARD
TO
TAPE
UPDATE
TAPE
SORT UPDATE TAPE
MERGE WITH
OLD DICTIONARY
NEW
DICTIONARY
TAPE
OLD
DICTIONARY
TAPE
CREATE TRANSFER
ADDRESS MATRIX
PROCESS DICTIONARY
STORE PACKAGES
IN ARRAYS
PROCESSED
DICTIONARY
TAPE

THIS TAPE IS READ INTO MEMORY BY THE TEXT EDITING PROGRAM

The three separate programs shown above may be combined and run as one program.

FIG. 7

APPENDIX

1. Sample Output of Automatic Indexing Using Full Text as Input

38

BUTLER TC, KUROIWA Y, WADDELL WJ
EFFECTS OF 5,5-DIMETHYL-2,4-OXAZOLIDINEDIONE (DMO) ON ACID-BASE AND ELECTROLYTE EQUILIBRIA.
J PHARMACOL EXP THER 152,62-6, APR 66

NO. OF WORDS = 2398
127480 (3) TRIMETHADIONE, 3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE, OXAZOLIDINEDIONES

TRIMETHADIONE/ ADMINISTRATION (1), EFFECT (1), ACTIVITY (1), DOSAGE (2), EPILEPSY (1), THERAPY (1)

END OF ARTICLE

SWINYARD EA, CASTELLION AW
ANTICONVULSANT PROPERTIES OF SOME BENZODIAZEPINES.
J PHARMACOL EXP THER 151,369-75, MAR 66

NO. OF WORDS = 3339
(3) ANTICONVULSANTS

127480 (3) TRIMETHADIONE, 3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE, OXAZOLIDINEDIONES

50066 (1) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

ANTICONVULSANTS/ PROPERTIES (1), ACTIVITY (2), EFFECT (2), EPILEPSY (1)
TRIMETHADIONE/ ADMINISTRATION (2), DOSAGE (1), ACTIVITY (1)
PHENOBARBITAL/ ACTIVITY (1)

END OF ARTICLE

CHEN G, ENSOR CR, BOHNER B
THE NEUROPHARMACOLOGY OF
2-(OMICRON-CHLOROPHENYL)-2-METHYLAMINOCYCLOHEXANONE HYDROCHLORIDE.
J PHARMACOL EXP THER 152,332-9, MAY 66

NO. OF WORDS = 3050
57330 (1) PENTOBARBITAL SODIUM,
SODIUM 5-ETHYL-5-(1-METHYLBUTYL)BARBITURATE, BARBITURATES

(2) ANTICONVULSANTS

ANTICONVULSANTS/ ADMINISTRATION (1), ACTIVITY (1), DOSAGE (1),
EFFECT (1)

END OF ARTICLE

CAMPBELL FG, GRAHAM JG, ZILKHA KJ
CLINICAL TRIAL OF CARBAZEPINE (TEGRETOL) IN TRIGEMINAL NEURALGIA.
J NEUROL NEUROSURG PSYCHIAT 29,265-7, JUN 66

NO. OF WORDS = 1831
298464 (1) CARBAMAZEPINE, TEGRETOL

(1) ANTICONVULSANTS

ANTICONVULSANTS/ EFFECT (1)

END OF ARTICLE

COOPER P
DRUGS FOR CONVULSIVE DISORDERS.
MIDWIFE HEALTH VISIT 2,212-3, MAY 66

NO. OF WORDS = 1340
(3) ANTICONVULSANTS

50066 (3) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

76948 (1) 5-METHYL-5-PHENYLBARBITURIC ACID, BARBITURATES

125337 (3) PRIMIDONE, 5-PHENYL-5-ETHYLHEXAHYDROPYRIMIDINE-4,6-DIONE, BARBITURATES

630933 (2) DIPHENYLHYDANTOIN SODIUM, SODIUM 5,5-DIPHENYL HYDANTOINATE, HYDANTOINS, GAROIN

57410 (3) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE, HYDANTOINS

50124 (2) MEPHENYTOIN, 5-ETHYL-3-METHYL-5-PHENYLHYDANTOIN, HYDANTOINS, METHOIN, MESONTOIN

127480 (3) TRIMETHADIONE, 3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE, OXAZOLIDINEDIONES, TROXIDONE, TRIDIONE

77678 (1) ETHOSUXIMIDE, 2-ETHYL,2-METHYLSUCCINIMIDE, SUCCINIMIDES

86340 (1) PHENSUXIMIDE, N-METHYL-2-PHENYLSUCCINIMIDE, SUCCINIMIDES

77418 (1) CELONTIN, N,2-DIMETHYL-2-PHENYLSUCCINIMIDE, SUCCINIMIDES, METHSUXIMIDE

501688 (2) N-BENZYL-BETA-CHLOROPROPIONAMIDE, CHLOROETHYLPHENAMIDE, BECLAMIDE

61563 (1) TETRAHYDRO-2-P-SULFAMOYL-PHENYL-1,2-THIAZINE-1,1-DIOXIDE, SULTHIAME

90493 (2) (2-PHENYLBUTYRYL)UREA, PHENETURIDE

63989 (1) PHENACEMIDE, PHENYLACETYLUREA

298464 (1) CARBAMAZEPINE, TEGRETOL

ANTICONVULSANTS/ THERAPY (1), DOSAGE (1), EFFECT (2), PROPERTIES (1)
PHENYLMETHYLBARBITURIC ACID/ DOSAGE (1)
BARBITURATES/ DOSAGE (2)
PHENOBARBITONE/ DOSAGE (2), EFFECT (1)
PRIMIDONE/ EFFECT (1), DOSAGE (1)
PHENYTOIN/ DOSAGE (1), THERAPY (1)
TROXIDONE/ EFFECT (1)
BECLAMIDE/ EFFECT (1)

END OF ARTICLE

SMITH DL, KEASLING HH, FORIST AA
THE METABOLISM OF N-ALKYL-4-BROMOBENZENESULFONAMIDES IN THE MOUSE. CORRELATION WITH ANTICONVULSANT ACTIVITY.
J MED CHEM 8,520-4, JUL 65

NO. OF WORDS = 3014
(3) ANTICONVULSANTS

ANTICONVULSANTS/ ACTIVITY (3), EFFECT (2), ADMINISTRATION (1), THERAPY (1)

END OF ARTICLE

44

LOWRY OH
METABOLITE LEVELS AS INDICATORS OF CONTROL MECHANISMS.
FED PROC 25,846-9, MAY-JUN 66

NO. OF WORDS = 1616
50066 (1) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

END OF ARTICLE

KLEIN JP
DIPHENYLHYDANTOIN INTOXICATION ASSOCIATED WITH HYPERGLYCEMIA.
J PEDIAT 69,463-5, SEP 66

NO. OF WORDS = 1219
57410 (3) DIPHENYLHYDANTOIN, 5,5-CIPHENYL-2,4-IMIDAZOLIDINEDIONE,
HYDANTOINS

50066 (2) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENCBARBITAL, BARBITURATES

(1) ANTICONVULSANTS

DIPHENYLHYDANTOIN/ ADMINISTRATION (2), EFFECT (1), DOSAGE (1)
ANTICONVULSANTS/ ADMINISTRATICN (1)

END OF ARTICLE

46

DOUGLAS A, SIMPSON D, MERCHANT S
THE EFFECT OF ANTISPASMODIC DRUGS ON THE ENDOMURAL BRONCHIAL (OR
SQUEEZE) PRESSURES IN BRONCHITIS AND ASTHMA.
AMER REV RESP DIS 93,703-15, MAY 66

NO. OF WORDS = 4425
(3) ANTISPASMODIC

ANTISPASMODIC/ EFFECT (2), ADMINISTRATION (1), ACTIVITY (1)

END OF ARTICLE

LIPP JA
EPILEPSY. II. BASIC NEUROPHYSIOLOGICAL ASPECTS OF ANTICONVULSANT DRUGS.
APPL THER 8, 437-41, MAY 66

NO. OF WORDS = 3001
(3) ANTICONVULSANTS

(3) BARBITURATES

50066 (2) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

115388 (1) MEPHOBARBITAL, 5-ETHYL-1-METHYL-5-PHENYLBARBITURIC ACID, BARBITURATES

125337 (2) PRIMIDONE, 5-PHENYL-5-ETHYLHEXAHYDROPYRIMIDINE-4,6-DIONE, BARBITURATES

(1) HYDANTOINS

57410 (3) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE, HYDANTOINS

127480 (2) TRIMETHADIONE, 3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE, OXAZOLIDINEDIONES

86340 (1) PHENSUXIMIDE, N-METHYL-2-PHENYLSUCCINIMIDE, SUCCINIMIDES

77418 (1) CELONTIN, N,2-DIMETHYL-2-PHENYLSUCCINIMIDE, SUCCINIMIDES, METHSUXIMIDE

77678 (1) ETHOSUXIMIDE, 2-ETHYL,2-METHYLSUCCINIMIDE, SUCCINIMIDES

63989 (1) PHENACEMIDE, PHENYLACETYLUREA

50124 (1) MEPHENYTOIN, 5-ETHYL-3-METHYL-5-PHENYLHYDANTOIN, HYDANTOINS

ANTICONVULSANTS/ EPILEPSY (1), ACTIVITY (3), EFFECT (1), PROPERTIES (1), THERAPY (1)
BARBITURATES/ EPILEPSY (1), ACTIVITY (2), EFFECT (1)
HYDANTOINS/ EPILEPSY (1)
DIPHENYLHYDANTOIN/ ACTIVITY (2), THERAPY (1), EFFECT (1), EPILEPSY (1)
TRIMETHADIONE/ ADMINISTRATION (1), ACTIVITY (2), DOSAGE (1)
PRIMIDONE/ EPILEPSY (1), THERAPY (1)
PHENOBARBITAL/ ACTIVITY (1), EPILEPSY (1), DOSAGE (1)
MEPHOBARBITAL/ THERAPY (1)

END OF ARTICLE

48

NEALON TF JR, SUGERMAN H, SHEA W
AN EXTRACORPOREAL DEVICE TO TREAT BARBITURATE POISONING. USE OF ANION- EXCHANGE RESINS IN DOGS.
JAMA 197, 158-160, 11 JUL 66

NO. OF WORDS = 1639
(3) BARBITURATES

50066 (3) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

PHENOBARBITAL/ THERAPY (3), DOSAGE (2)
BARBITURATES/ ADMINISTRATION (1), THERAPY (1)

END OF ARTICLE

COLLINS VJ
ADVANCES IN PHARMACOLOGY RELATED TO ANESTHESIA AND SURGERY.
INDUSTR MED SURG 35,465-71, JUN 66

NO. OF WORDS = 4594
(2) BARBITURATES

(1) ANTISPASMODIC

BARBITURATES/ THERAPY (1), ACTIVITY (1), EFFECT (1), DOSAGE (1)

END OF ARTICLE

50

BERNSTEIN ZL
TREATMENT OF BARBITURATE COMA.
NEW YORK J MED 66, 2290-4, 1 SEP 66

NO. OF WORDS = 2428
57410 (2) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE,
HYDANTOINS

(3) BARBITURATES

57432 (1) AMOBARBITAL, 5-ETHYL-5-ISOAMYLBARBITURIC ACID, BARBITURATES

50066 (1) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

DIPHENYLHYDANTOIN/ THERAPY (1)
BARBITURATES/ EFFECT (1), ACTIVITY (1), THERAPY (3), DOSAGE (1)

END OF ARTICLE

CHINITZ A, SEELINGER DF, GREENHOUSE AH
ANTICONVULSANT THERAPY IN TRIGEMINAL NEURALGIA.
AMER J MED SCI 252,62-7, JUL 66

NO. OF WORDS = 2427
630933 (3) DIPHENYLHYDANTOIN SODIUM, SODIUM 5,5-DIPHENYL HYDANTOINATE, HYDANTOINS

(2) ANTICONVULSANTS

298464 (3) CARBAMAZEPINE, TEGRETOL

DIPHENYLHYDANTOIN SODIUM/ EFFECT (3), THERAPY (2), ADMINISTRATION (1), PROPERTIES (1), ACTIVITY (1)
ANTICONVULSANTS/ EFFECT (1), PROPERTIES (1)
CARBAMAZEPINE/ THERAPY (1), EFFECT (1)

END OF ARTICLE

ARMBRECHT EC, STEERE DW
A CASE REPORT OF EVALUATION AND MANAGEMENT OF HYPERPLASTIC GINGIVA IN EPILEPTICS CAUSED BY SODIUM DILANTIN THERAPY.
W VIRGINIA DENT J 40,43-4, JUL 66

NO. OF WORDS = 906
50066 (2) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

630933 (2) DIPHENYLHYDANTOIN SODIUM, SODIUM 5,5-DIPHENYL HYDANTOINATE, HYDANTOINS, DILANTIN SODIUM

(2) ANTICONVULSANTS

(2) BARBITURATES

57410 (2) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE, HYDANTOINS, DILANTIN

DILANTIN SODIUM/ THERAPY (2)
ANTICONVULSANTS/ EPILEPSY (2)
BARBITURATES/ EPILEPSY (2)
DILANTIN/ EFFECT (2), THERAPY (2)

END OF ARTICLE

GILBERT JC, ORTIZ WR, MILLICHAP JG
THE EFFECTS OF ANTICONVULSANT DRUGS ON THE PERMEABILITY OF BRAIN CELLS TO D-XYLOSE.
J NEUROCHEM 13,247-55, APR 66

NO. OF WORDS = 3338
(3) ANTICONVULSANTS

50066 (3) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

695534 (3) DIMETHADIONE, 5,5-DIMETHYLOXAZOLIDINE-2,4-DIONE, OXAZOLIDINEDIONES

57410 (3) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE, HYDANTOINS

127480 (1) TRIMETHADIONE, 3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE, OXAZOLIDINEDIONES

ANTICONVULSANTS/ EFFECT (1), ACTIVITY (1)
PHENOBARBITONE/ EFFECT (2), THERAPY (1), ADMINISTRATION (1), ACTIVITY (1)
DIPHENYLHYDANTOIN/ ACTIVITY (1), EFFECT (2)
DIMETHADIONE/ ACTIVITY (1)
TRIMETHADIONE/ THERAPY (1)

END OF ARTICLE

54

HUDGINS RL, CORBIN KB
AN UNCOMMON SEIZURE DISORDER, FAMILIAL PAROXYSMAL CHOREOATHETOSIS.
BRAIN 89,199-204, JUN 66

NO. OF WORDS = 2498
115388 (1) MEPHOBARBITAL, 5-ETHYL-1-METHYL-5-PHENYLBARBITURIC ACID, BARBITURATES

57410 (2) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE, HYDANTOINS

(2) ANTICONVULSANTS

50066 (1) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

ANTICONVULSANTS/ THERAPY (2)
PHENOBARBITAL/ THERAPY (1)

END OF ARTICLE

2. Automatic Indexing Program Listing

```
71              BAXENDA                       FORTRAN SOURCE LIST
 ISN        SOURCE STATEMENT

   0 $IBFTC EDIT
   1        DIMENSION INBUF(72)
   2        DIMENSION NMWORD(50,2)
   3        DIMENSION INDIC(3)
   4        DATA BLANK,Z,COMMA,PERIOD/1H ,1HZ,1H,,1H./
   5        DATA INDIC/1H1,1H2,1H3/
   6        DATA MA,X,DASH,LP,RP,SLASH,AST,EIGHT/1HM,1HX,1H-,1H(,1H),1H/,1H*,
           11H8/
   7        COMMON /H/ MARGIN
  10        COMMON /LIN/ LINE(132),LINDEX,LMARG,DICT(3000)
  11        COMMON KWORD(14),NWRDS,WORD(72),LENGTH
  12        INTEGER DICT
  13        INTEGER PACK4(350),PACK1(100)
  14        INTEGER COLUMN,PERIOD,COMMA,Z,BLANK,WORD,TRADD(4096)
  15        INTEGER GOODWD(100,2),GINDEX
  16        INTEGER SAVE,SVINDX,SUPL(750)
  17        INTEGER SVLNGH
  20        INTEGER SINDX2
  21        INTEGER SUPL2(400)
  22        INTEGER PINDEX,PACKNO,REGNO
  23        INTEGER ENDSEN
  24        INTEGER SUPL3(50,3),TYPE1,TYPE2,SINDX3
  25        INTEGER SW1,SW2
  26        INTEGER WRDCT
  27        INTEGER PTERM
  30        INTEGER RP,X,DASH,EIGHT,SLASH,AST
  31        INTEGER PCOUNT
     C ****************************************************
     C ****** DATA INPUT AND INITIALIZATION SECTION ******
     C ****************************************************
  32        REWIND 2
     C READ IN TRANSFER ADDRESS ARRAY FROM TAPE B-3.
  33        READ (2) TRADD
     C READ IN DICTIONARY SIZE AND CONDENSED DICTIONARY.
  35        READ (2) J,(DICT(I),I=1,J)
     C READ IN PACKAGE ARRAYS.
  43        READ(2) PACK4,PACK1
  46        REWIND 2
     C READ IN THE PARAMETER CARD.
     C IT IS REQUIRED TO HAVE NUMBERS PUNCHED IN COLUMNS 1-3, 5, 7, AND 8-10
     C OF THE PARAMETER CARD (THE FIRST CARD TO BE READ).
     C THE THREE DIGIT NUMBER PUNCHED IN COLS. 1-3 IS USED BY THE SUBROUTINE
     C     -HEAD- FOR THE MARGIN WIDTH OF THE HEADING OF EACH ARTICLE.
     C A NUMBER PUNCHED IN COLUMN 5 HAS THE FOLLOWING MEANING
     C          1...LIST COMPARISONS AND LARGE WORDS.
     C          2...LIST COMPARISONS ONLY.
     C          3...LIST LARGE WORDS ONLY.
     C          4...DO NOT LIST COMPARISONS OR LARGE WORDS.
     C A NUMBER PUNCHED IN COLUMN 7 HAS THE FOLLOWING MEANING
     C          1...SAVE NON-MEDICAL TERMS.
     C          2...DO NOT SAVE NON-MEDICAL TERMS. (THEIR ASSOCIATION WITH
     C                MEDICAL TERMS WILL BE SAVED).
     C THE THREE DIGIT NUMBER PUNCHED IN COLS. 8-10 IS FOR USE BY THE
     C    SUBROUTINE -WRDPUT- FOR THE WIDTH OF THE PRINT LINE USED FOR
     C    OUTPUT OF VALID MEDICAL WORDS, NON-MEDICAL WORDS, AND ASSOCIATIONS.
```

```
ISN       SOURCE STATEMENT

 47          READ (5,1) MARGIN,SW1,SW2,LMARG
 54        1 FORMAT (I3,2I2,I3)
       C INITIALIZE LENGTH
 55        2 LENGTH = 0
       C INITIALIZE WORC COUNT TO 0.
 56          WRDCT = 0
       C SET -VALID WORC- INDICATOR TO 0 (OFF).
 57          IVALID=0
       C SET -PREVICUS VALID WORD IN SENTENCE- INDICATOR TO 0 (OFF).
 60          ISENT=C
       C SET -REACHED END OF SENTENCE- INDICATOR TO 0 (OFF).
 61          ENDSEN=0
       C INITIALIZE INDEXES FOR 5 FILES.
 62          GINDEX=1
 63          NMINDX = 1
 64          SVINDX=1
 65          SINDX2=1
 66          SINDX3=1
       C SET FIRST WORD IN EACH FILE TO 0 (THIS IDENTIFIES THE -END OF FILE-
       C    FOR EACH FILE).
 67          GOODWD(1,1)=0
 70          NMWORD(1,1) = 0
 71          SUPL(1)=0
 72          SUPL2(1)=0
 73          SUPL3(1,1)=0
 74          SUPL3(1,2)=0
       C PROCESS THE BIBLIOGRAPHIC CITATION FOR THE NEXT ARTICLE.
 75          CALL HEAD
       C ***************************************
       C ****** WORC EXTRACTION SECTION ******
       C ***************************************
       C READ IN ONE CARD CF THE TEXT.
 76        3 READ (5,4) INBUF
100        4 FORMAT (72A1)
       C CHECK FOR END OF ARTICLE CARD (ZZZ)
101          IF (INBUF(1).EQ.Z.AND.INBUF(2).EQ.Z.AND.INBUF(3).EC.Z) GO TO 44
       C INITIALIZE TO CHECK BEGINNING OF INPUT BUFFER.
104          DO 5 I=1,72
       C LOOK FOR A BLANK BETWEEN TWO WORDS.
105          IF (INBUF(I).EQ.BLANK) GO TO 6
110          LENGTH=LENGTH+1
       C REMOVE THE NON-BLANK CHARACTER FROM THE INPUT BUFFER.
111          WORD(LENGTH)=INBUF(I)
112        5 CONTINLE
114          GO TO 3
       C CHECK TO SEE IF WORD IS NULL.
115        6 IF(LENGTH.EQ.0) GO TO 5
       C INCREASE THE WCRD COUNT.
120          WRDCT = WRDCT+1
       C WORD IS NOT NULL, CHECK FOR PUNCTUATION AND END CF SENTENCE.
121          IF (WORD(LENGTH).EQ.PERIOD) ENDSEN=1
124          IF (WORD(LENGTH).EQ.PERIOD.OR.WORD(LENGTH).EC.COMMA) GO TO 7
       C THERE IS NC PUNCTUATION
127          GO TO 8
       C REMOVE LAST CHARACTER (PUNCTUATION)
```

```
            BAXENDA                       FORTRAN SOURCE LIST EDIT
ISN        SOURCE STATEMENT

130       7 WORD(LENGTH)=BLANK
131         LENGTH=LENGTH-1
      C CHECK FOR A SHORT WORD
132       8 IF(LENGTH.LT.4) GO TO 25
      C CONDENSE THE TEXT WORD TO A6 FORMAT.
135         CALL COMPOS
      C ****************************************
      C ****** DICTIONARY SEARCH SECTION ******
      C ****************************************
      C GENERATE TRANSFER MATRIX ADDRESS INDEX
136         IF (KWORD(1).LT.0) GO TO 9
141         INDEX=KWORD(1)/2**24
142         GO TO 10
143       9 INDEX=2**11-KWORD(1)/2**24
144      10 INDEX=INDEX+1
145         M = TRADD(INDEX)
      C IF TRANSFER ADDRESS IS 0, WORD IS NO GOOD
146      11 IF (M.EQ.0) GO TO 22
      C IF WORD IS ALPHABETICALLY BELOW DICTIONARY ENTRY, END DICTIONARY
      C SEARCH
151         IF (KWORD(1)) 12,25,13
152      12 IF (KWORD(1).GT.DICT(M)) GO TO 19
155         GO TO 14
156      13 IF (DICT(M).GT.KWORD(1)) GO TO 19
161      14 MTEMP=M
      C CHECK FOR A MAXIMUM COMPARISON WITH A DICTIONARY WORD
162         DO 15 MAXCOM=1,NWRDS
163         IF (DICT(MTEMP).NE.KWORD(MAXCOM)) GO TO 18
166      15 MTEMP=MTEMP+1
      C TEST TEXT WORD AND DICT WORD FOR EQUAL LENGTH, IF SAME ... GOOD WORD
170         IF (NWRDS.EQ.(DICT(M-2)-(M+2)))GO TO 28
173         SAVE=M
174         MAX=1
      C MODIFY INDEX M UP TO NEXT DICT ENTRY.
175      16 M=DICT(M-2)
176         GO TO 11
      C *******************************************
      C ****** MULTIPLE WORD CHECK SECTION *******
      C *******************************************
      C INCREASE LENGTH AND INDEX I
177      17 LENGTH = LENGTH+1
200         I=I+1
      C PUT IN A BLANK
201         WORD(LENGTH)=BLANK
202         SVLNGH=LENGTH+1
      C SET INDICATOR TO SHOW THAT A MULTIPLE WORD CHECK IS IN PROGRESS.
203         MULWRD=1
      C GO BACK TO WORD EXTRACTION SECTION
204         GO TO 5
      C ****************************************************
      C ****** SAVE WORD SECTION (PARTIAL COMPARISON) ******
      C ****************************************************
      C LOOK FOR PARTIAL COMPARISON, INDEX MAXCOM=1 MEANS NO COMPARISON
205      18 IF ((MAXCOM.EQ.1).OR.(MAXCOM.LT.MAX+1)) GO TO 16
      C SET SAVE INDICATOR TO DICTIONARY INDEX
```

```
ISN       SOURCE STATEMENT

210           SAVE = M
      C SET MAX TO MAXIMUM COMPARISON
211           MAX=MAXCOM-1
212           GO TO 16
      C TEST SAVE INDICATOR FOR A NON-VALID WORD, IF ZERO, DO NOT SAVE
      C GO TEST FOR LENGTH
213        19 IF (SAVE.EQ.0) GO TO 22
      C IF MULTIPLE WORD CHECK IS IN PROGRESS DO NOT SAVE ANY PARTIAL COMPARISONS
216           IF (MULWRD.EQ.1) GO TO 25
      C WE NOW HAVE A SINGLE WORD THAT COMPARES TO A WORD IN THE DICTIONARY.
      C TEST PROGRAM PARAMETER TO SEE IF THIS WORD SHOULD BE SAVED.
221           GO TO (20,20,17,17), SW1
      C PUT DICTIONARY INDEX, MAXIMUM COMPARISON, WORD LENGTH, AND TEXT WORD
      C IN SUPPLEMENTARY RECORD
222        20 SUPL(SVINDX) = SAVE
223           SUPL(SVINDX+1)=MAX
224           SUPL(SVINDX+2)=NWRDS
225           SVINDX=SVINDX+3
226           DO 21 J=1,NWRDS
227           SUPL(SVINDX)=KWORD(J)
230        21 SVINDX=SVINDX+1
232           SUPL(SVINDX)=0
      C ZERO WILL BE WRITTEN OVER BY NEXT ENTRY IN SUPL. RECORD, OTHERWISE
      C IT IS A SIGNAL FOR THE END OF THE SUPL. RECORD
233           GO TO 17
      C ********************************************
      C ****** SAVE WORD SECTION (LARGE WORD) ******
      C ********************************************
      C TEST FOR LARGE SINGLE WORDS.
234        22 IF ((LENGTH.LT.18).OR.(MULWRD.EQ.1)) GO TO 25
      C PUNCH A CARD WITH THE LONG WORD.
237           WRITE (7,80) (KWORD(J),J=1,NWRDS)
      C WE NOW HAVE A LARGE WORD (18 OR MORE CHARACTERS).  TEST PROGRAM
      C       PARAMETER TO SEE IF THIS WORD SHOULD BE SAVED.
244           GO TO (23,25,23,25), SW1
      C SAVE WORD IN SUPL. RECORD 2
245        23 SUPL2(SINDX2) = NWRDS
246           DO 24 J = 1,NWRDS
247           SINDX2=SINDX2+1
250        24 SUPL2(SINDX2)=KWORD(J)
252           SINDX2=SINDX2+1
253           SUPL2(SINDX2)=0
      C ZERO WILL BE WRITTEN OVER BY NEXT ENTRY, OTHERWISE IT IS AN END OF
      C        RECORD SIGNAL.
      C ******************************************
      C ****** WORD CHECK COMPLETED SECTION ******
      C ******************************************
      C CHECK FOR A MULTIPLE WORD.
254        25 IF (MULWRD.EQ.0) GO TO 27
      C IF THIS WORD IS VALID, GO TO VALID WORD SECTION.
257           IF (IVALID.NE.0) GO TO 29
      C RESTORE SECOND (ADDENDED WORD) OF MULTIPLE WORD FOR A SEPARATE CHECK.
262           J=1
263           DO 26 K=SVLNGH,LENGTH
264           WORD(J)=WORD(K)
```

```
71                BAXENCA                         FORTRAN SOURCE LIST EDIT
      ISN        SOURCE STATEMENT

      265     26 J=J+1
      267        LENGTH=LENGTH+1-SVLNGH
      270        MULWRD=0
      271        SAVE=0
          C GO BACK TO PROCESS THIS WORD.
      272        GO TO 8
          C REINITIALIZE FCR A NEW TEXT WORD
      273     27 LENGTH=0
      274        SAVE=0
      275        MAX=0
          C CHECK TO SEE IF END OF SENTENCE WAS REACHED.  IF SC, TURN OFF
          C      -PREVICUS VALID WORD IN SENTENCE- INDICATOR.
      276        IF (ENCSEN.EQ.1) ISENT=0
      301        ENDSEN=0
      302        GO TO 5
          C ********************************
          C ******VALIC WORD SECTION ******
          C ********************************
      303     28 IVALID=M
          C EVEN THOUGH A WORD IS VALID, IT IS POSSIBLE FOR IT TO BE PART OF A
          C MULTIPLE WCRC, SO TEST TO SEE IF THIS VALID WORD IS A SINGLE WCRD CR
          C    A MULTIPLE WORD.
      304        IF (MULWRD.EQ.1) GC TO 29
          C GO PICK UP SECCND WORD FROM TEXT.
      307        GO TO 17
          C IF THERE HAS BEEN ANOTHER VALID WORD IN THE SAME SENTENCE, GO TO
          C VALID WORD ASSCCIATION SECTION.
      310     29 IF (ISENT.NE.0) GO TO 38
          C PICK UP PACKAGE NUMBER OF THIS WORD FROM DICTICNARY.
      313     30 ITEMP = IABS(DICT(IVALID-1))
          C TEST PACKAGE NUMBER TO SEE IF WORD IS MEDICAL CR NCN-MEDICAL.
      314        IF (ITEMP.LT.200) GO TO 34
          C WORD IS NON-MEDICAL, TEST PROGRAM PARAMETER TO SEE IF IT SHOULD BE
          C       SAVED.
      317        GO TO (31,36), SW2
      320     31 IVALID = PACK1(ITEMP-200)
          C SEARCH NON-MEDICAL FILE FOR THE SAME WORD.
      321        DO 32 J = 1,NMINDX
      322        IF (IVALID.EQ.NMWORD(J,1)) GO TO 33
      325     32 CONTINLE
          C PUT NEW WORD AND THE COUNT IN THE FILE.
      327        NMWORD(NMINDX,1) = IVALID
      330        NMWORD(NMINDX,2) = 1
      331        NMINDX = NMINDX+1
          C SET UP A NEW END OF FILE INDICATOR.
      332        NMWORD(NMINDX,1) = 0
      333        GO TO 36
          C SINCE WORD IS ALREADY IN THE FILE, JUST INCREASE ITS COUNT.
      334     33 NMWORD(J,2) = NMWORD(J,2)+1
      335        GO TO 36
          C PRCCESS THE VALID MEDICAL WORD, SEARCH THE FILE FOR THE SAME WCRD.
      336     34 DO 35 J=1,GINDEX
      337        IF (IVALID.EQ.GOODWD(J,1)) GO TO 37
      342     35 CONTINLE
          C PUT NEW WORD INTO LIST OF VALID MEDICAL WORDS.
```

```
ISN        SOURCE STATEMENT

344           GOCDWD(GINCEX,1)=IVALID
345           GOCDWD(GINCEX,2)=1
346           GINDEX=GINCEX+1
          C THE ZERO AT THE END OF THE LIST IS AN END OF LIST SIGNAL
347           GOODWD(GINCEX,1)=0
          C SET INDICATOR SHOWING A VALID WORD IN SENTENCE BEING PROCESSED.
350        36 ISENT=IVALID
351           IVALID=0
          C GO PICK UP A NEW TEXT WORD
352           GO TO 25
          C SINCE WORD IS ALREADY IN LIST, ONLY INCREASE ITS COUNT.
353        37 GOODWD(J,2)=GOODWD(J,2)+1
354           GO TO 36
          C ********************************************
          C ****** VALID WCRD ASSOCIATIONS SECTION ******
          C ********************************************
          C EXAMINE THE PRESENT WORD TO SEE IF IT IS MEDICAL OR NON-MEDICAL.
355        38 TYPE1 = (IABS(DICT(IVALID-1))+99)/100
356           GO TO (39,39,43), TYPE1
          C THE PRESENT WORD IS A MEDICAL TERM.
357        39 TYPE2 = (IABS(DICT(ISENT-1))+99)/100
          C TEST THE PREVICUS VALID WORD IN SENTENCE TO SEE IF IT IS A NONMEDICAL
          C     WORD.
360           IF (TYPE2.NE.3) GO TO 30
363           IVAL1=IVALID
364           IVAL2=ISENT
          C CONVERT THE NON-MEDICAL WORD TO ITS BASIC TERM.
365        40 IVAL2 = IABS(DICT(IVAL2-1))-200
366           IVAL2 = PACK1(IVAL2)
367           PACKNO = IABS(DICT(IVAL1-1))
          C CONVERT, IF NECESSARY, THE MEDICAL WORD TO ITS BASIC TERM.
370           IF (PACKNO.EQ.50.OR. PACKNO .EQ.49)IVAL1=PACK4(4*PACKNO-2)
          C SEARCH THE LIST OF ASSOCIATED VALID WORDS FOR THE SAME TWO WORDS.
373           DO 41 J=1,SINDX3
374           IF (IVAL1.EQ.SUPL3(J,1).AND.IVAL2.EQ.SUPL3(J,2)) GO TO 42
377        41 CONTINLE
          C PUT THE TWC NEW TERMS AT THE END OF THE LIST.
401           SUPL3(SINDX3,1)=IVAL1
402           SUPL3(SINDX3,2)=IVAL2
          C PUT IN A CCUNT OF THE ASSOCIATION.
403           SUPL3(SINDX3,3)=1
404           SINDX3=SINCX3+1
          C SET UP AN -END OF LIST- INDICATOR.
405           SUPL3(SINDX3,1)=0
406           SUPL3(SINDX3,2)=0
407           GO TO 30
          C THE TWO TERMS ARE ALREADY IN THE LIST, INCREASE THE COUNT.
410        42 SUPL3(J,3) = SUPL3(J,3)+1
411           GO TO 30
          C THE PRESENT VALID WORD IS NON-MEDICAL, TEST THE PREVIOUS VALID WORD
          C IN SENTENCE TO SEE IF IT IS A MEDICAL TERM.
412        43 TYPE2=TYPE1
413           TYPE1=(IABS(DICT(ISENT-1))+99)/100+1
414           IF (TYPE1.NE.1.AND.TYPE1.NE.2) GO TO 30
417           IVAL1=ISENT
```

```
ISN        SOURCE STATEMENT

420            IVAL2=IVALID
421            GO TO 40
         C ************************************
         C ****** END OF ARTICLE SECTION ******
         C ************************************
422         44 GINDEX = GINDEX-1
         C PRINT THE NUMBER OF WORDS IN THIS ARTICLE.
423            WRITE (6,45) WRDCT
424         45 FORMAT (15HONO. OF WORDS =,I5)
         C      **********************************
         C      * PRINT VALID WORDS SUB-SECTION *
         C      **********************************
         C PROCESS LIST OF VALID WORDS.
425            DO 60 I = 1,GINDEX
         C TEST TO SEE IF THIS WORD WAS ALREADY DONE.
426            IF (GOODWD(I,1).EQ.0) GO TO 60
431            IVALID = GOODWD(I,1)
         C INITALIZE PRINT LINE INDEX.
432            LINDEX = 0
433            PACKNO = IABS(DICT(IVALID-1))
         C LOCATE THE PACKAGE FOR THIS WORD.
434            PINDEX = PACKNO*4-3
         C GET THE REGISTRY NUMBER OUT OF THE PACKAGE.
435            REGNO = PACK4(PINDEX)
         C TEST FOR A NULL REGISTRY NUMBER OR ONE WITH AN MX PREFIX.
436            IF (REGNO) 47,51,46
437         46 ITEMP = REGNO
440            GO TO 48
441         47 ITEMP = -(REGNO+8000000)
         C PLACE MX-8 PREFIX IN LINE.
442            LINE(1) = MA
443            LINE(2) = X
444            LINE(3) = DASH
445            LINE(4) = EIGHT
446            LINDEX = 4
         C CONVERT THE REGISTRY NUMBER (IN BINARY) TO CODED (A6 FORMAT).
447         48 KWORD(1) = ITEMP
450            CALL CONCOD(KWORD(1))
         C CONVERT THE A6 FORMAT TO A1.
451            NWRDS = 1
452            CALL DECOMP
         C LOOK FOR A LEADING ZERO IN THE REGISTRY NUMBER.
453            IF (ITEMP.LT.100000.AND.REGNO.GT.0) GO TO 49
456            LINDEX = LINDEX+1
         C PUT THE FIRST DIGIT IN THE PRINT LINE.
457            LINE(LINDEX) = WORD(1)
         C PLACE LAST 5 DIGITS OF REGISTRY NUMBER IN PRINT LINE.
460         49 DO 50 J = 2,6
461            LINDEX = LINDEX+1
462         50 LINE(LINDEX) = WORD(J)
464            LINDEX = LINDEX+1
465            LINE(LINDEX) = BLANK
         C INITIALIZE COUNT OF REFERENCES TO THIS PACKAGE.
466         51 PCOUNT = 0
         C LOOK THROUGH THE REST OF THE LIST FOR A REFERENCE TO THE SAME PACKAGE,
ISN       SOURCE STATEMENT

        C     IF FOUNC, INCREASE COUNT.
 467            DO 52 J = I,GINDEX
 470            IF (GOCDWD(J,1).EQ.0) GO TO 52
 473            ITEMP = GOCDWD(J,1)
 474            IF (IABS(DICT(ITEMP-1)).EQ.PACKNO) PCOUNT = PCOUNT+GOODWD(J,2)
 477         52 CONTINLE
        C RECORD THE NUMBER OF REFERENCES IN PRINT LINE.
 501            LINDEX = LINDEX+1
 502            LINE(LINDEX) = LP
 503            LINDEX = LINDEX+1
        C REDUCE THE PACKAGE REFERENCE CCUNT TO AN INDICATOR.
 504            IN = 2
 505            IF (PCCUNT*1000.LE.WRDCT) IN = 1
 510            IF (PCCUNT*1000.GE.3*WRDCT) IN = 3
 513         53 LINE(LINDEX) = INDIC(IN)
 514            LINDEX = LINDEX+1
 515            LINE(LINDEX) = RP
 516            LINDEX = LINDEX+1
 517            LINE(LINDEX) = BLANK
        C NOW PROCESS EACH TERM CONTAINED IN THE PACKAGE.
 520            DO 54 J = 1,3
 521            JJ = PINDEX+J
 522            PTERM = PACK4(JJ)
        C CHECK FOR A NULL TERM IN THE PACKAGE.
 523            IF (PTERM.EQ.0) GO TO 54
        C PLACE THE PACKAGE TERM IN THE PRINT LINE.
 526            CALL WRDPUT(PTERM,2)
 527            LINDEX = LINDEX+1
 530            LINE(LINDEX) = COMMA
 531            LINDEX = LINDEX+1
 532            LINE(LINDEX) = BLANK
        C EXAMINE THE REST OF THE VALIC WORD LIST FOR THIS SAME PACKAGE TERM.
 533            DO 54 K = I,GINDEX
 534            IF (GOCDWD(K,1).NE.PTERM) GO TC 54
        C STRIKE THIS WORD FROM THE LIST SINCE IT HAS BEEN PRINTED ALREADY.
 537            GOODWD(K,1) = 0
 540         54 CONTINLE
        C NOW EXAMINE THE LIST OF VALID WORDS FOR A WORD THAT BELONGS TO THE
        C     PACKAGE BEING PROCESSED BUT IS NOT CONTAINEC IN THE PACKAGE.
 543         55 DO 57 J= I,GINDEX
 544            IF (GOCDWD(J,1).EQ.0)  GO TO 57
 547            ITEST = GOCDWD(J,1)
 550            IF (IABS(DICT(ITEST-1)).NE.PACKNO) GO TO 57
        C WE HAVE FOLND A WORD BELONGING TO THE PACKAGE BEING PROCESSED.
 553            GOODWD(J,1) = 0
        C TEST THE PACKAGE NUMBER, IF NEGATIVE DO NOT PRINT THE WORD.
 554            IF (DICT(ITEST-1)) 57,57,56
        C PUT THE WORD IN THE PRINT LINE.
 555         56 CALL WRDPUT(ITEST,2)
 556            LINDEX = LINDEX+1
 557            LINE(LINDEX) = COMMA
 560            LINDEX = LINDEX+1
 561            LINE(LINDEX) = BLANK
 562         57 CONTINLE
        C CHECK FOR AN EMPTY PRINT LINE.
```

```
ISN        SOURCE STATEMENT

564          IF (LINDEX.EQ.0) GO TO 59
    C DO NOT PRINT LAST COMMA AND BLANK.
567          LINDEX =LINDEX-2
    C PRINT THE LAST LINE FOR THIS PACKAGE.
570          WRITE (6,58) (LINE(J),J=1,LINDEX)
575       58 FORMAT (1H ,132A1)
    C SKIP A LINE BETWEEN PACKAGES.
576       59 WRITE (6,58)
577       60 CONTINUE
    C        ***************************************
    C        * PRINT WORD ASSOCIATIONS SUB-SECTION *
    C        ***************************************
    C SKIP TWO LINES FOR NEXT OUTPUT SECTION.
601          WRITE (6,61)
602       61 FORMAT (1H0)
603          SINDX3=SINDX3-1
    C PROCESS LIST OF WORD ASSOCIATIONS.
604          DO 64 I = 1,SINDX3
    C SKIP ANY WORD ALREADY DONE.
605          IF (SUPL3(I,1).EQ.0) GO TO 64
610          M1 = SUPL3(I,1)
    C INITIALIZE FOR A NEW PRINT LINE.
611          LINDEX = 0
    C PUT THE MEDICAL TERM IN THE PRINT LINE FOLLOWED BY A SLASH.
612          CALL WRDPUT(M1,0)
613          LINDEX = LENGTH+1
614          LINE(LINDEX) = SLASH
    C SEARCH THE RECORD FOR THE SAME MEDICAL TERM.
615          DO 63 J = I,SINDX3
616          IF (SUPL3(J,1).NE.M1) GO TO 63
    C REMOVE THE MEDICAL TERM FROM THE FILE.
621          SUPL3(J,1) = 0
    C REMOVE THE NON-MEDICAL TERM.
622          M2 = SUPL3(J,2)
    C REMOVE THE COUNT FOR THIS ASSOCIATION.
623          NUMOCC = SUPL3(J,3)
624          LINDEX = LINDEX +1
625          LINE(LINDEX) = BLANK
    C PUT THE NON-MEDICAL TERM IN THE PRINT LINE.
626          CALL WRDPUT(M2,5)
    C INSERT A FREQUENCY COUNT INDICATOR.
627          LINDEX = LINDEX+1
630          LINE(LINDEX) = BLANK
631          LINDEX = LINDEX+1
632          LINE(LINDEX) = LP
633          LINDEX = LINDEX+1
634          IN = 2
635          IF (NUMOCC*1000.LE.WRDCT) IN = 1
640          IF (NUMOCC*1000.GE.3*WRDCT) IN = 3
643       62 LINE(LINDEX) = INDIC(IN)
644          LINDEX = LINDEX+1
645          LINE(LINDEX) = RP
646          LINDEX = LINDEX+1
647          LINE(LINDEX) = COMMA
650       63 CONTINUE
```

```
ISN       SOURCE STATEMENT

     C DO NOT PRINT LAST COMMA IN THE LINE.
652        LINDEX = LINDEX-1
     C PRINT THE INCOMPLETED LINE.
653        WRITE (6,58) (LINE(J),J=1,LINDEX)
660     64 CONTINUE
     C     ***************************************
     C     * PRINT NON-MEDICAL WORDS SUB-SECTION *
     C     ***************************************
     C SKIP TWO LINES BEFORE DOING NEXT PRINT SECTION.
662        WRITE (6,61)
     C TEST PROGRAM PARAMETER TO SEE IF NON-MEDICAL WORDS ARE TO BE LISTED.
663        GO TO (65,68) , SW2
664     65 NMINDX = NMINDX-1
     C CHECK FOR AN EMPTY FILE.
665        IF (NMINDX.EQ.0) GO TO 68
     C INITIALIZE PRINT LINE INDEX.
670        LINDEX = 0
     C PROCESS THE NON-MEDICAL FILE.
671        DO 67 I = 1,NMINDX
     C TAKE THE WORD AND FREQUENCY COUNT FROM THE FILE.
672        M = NMWORD(I,1)
673        NUMOCC = NMWORD(I,2)
     C PUT NON-MEDICAL TERM IN PRINT LINE.
674        CALL WRDPUT(M,6)
     C PUT THE NUMBER OF OCCURANCE OF TERMS IN LINE.
675        LINDEX = LINDEX+1
676        LINE(LINDEX) = LP
677        NDIGIT = 1
700        IF (NUMOCC.GT.9) NDIGIT = 2
     C CONVERT COUNT FROM BINARY TO CODED (A6 FORMAT).
703        CALL CONCOD(NUMOCC)
     C CONVERT FROM A6 TO A1 FORMAT.
704        KWORD(1) = NUMOCC
705        NWRDS =1
706        CALL DECOMP
707        DO 66 J =1,NDIGIT
710        JJ = 6+J-NDIGIT
711        LINDEX = LINDEX+1
712     66 LINE(LINDEX) = WORD(JJ)
714        LINDEX = LINDEX+1
715        LINE(LINDEX) = RP
716        LINDEX = LINDEX+1
717        LINE(LINDEX) = COMMA
720        LINDEX = LINDEX+1
721        LINE(LINDEX) = BLANK
722     67 CONTINUE
724        IF (LINDEX.EQ.0) GO TO 68
     C DO NOT PRINT THE LAST COMMA AND BLANK.
727        LINDEX = LINDEX-2
     C PRINT THE LAST INCOMPLETE LINE.
730        WRITE (6,58) (LINE(I),I=1,LINDEX)
     C     *********************************
     C     * PRINT COMPARISONS SUB-SECTION *
     C     *********************************
     C TEST PROGRAM PARAMETER TO SEE IF COMPARISONS ARE TO BE LISTED.
```

```
ISN       SOURCE STATEMENT

735     68 GO TO (69,69,74,78) , SW1
736     69 SVINDX = 1
      C PRINT HEADING.
737        WRITE (6,70)
740     70 FORMAT (1H0,15X,21HC O M P A R I S O N S/1H ,5X,6HDEGREE,5X,30HTEX
          1T WORD  -    DICTIONARY WORD)
      C CHECK FOR AN EMPTY FILE.
741        IF (SUPL(1).NE.0) GO TO 72
744        WRITE (6,71)
745     71 FORMAT (1H ,15X,7HN O N E)
746        GO TO 74
      C LOOK FOR THE -END OF FILE- INDICATOR.
747     72 IF (SUPL(SVINDX).EQ.0) GO TO 74
752        M=SUPL(SVINDX)
753        MAX=SUPL(SVINDX+1)
754        NWRDS=SUPL(SVINDX+2)
755        N=(DICT(M-2))-3
756        NWINDX=SVINDX+NWRDS+2
757        JSV=SVINDX+3
      C PRINT THE TEXT WORD AND DICTIONARY WORD SHOWING DEGREE OF COMPARISON.
760        WRITE (6,73) MAX,(SUPL(J),J=JSV,NWINDX),DASH,(DICT(K),K=M,N)
771     73 FORMAT (1H ,7X,I2,3X,29A6)
772        SVINDX=NWINDX+1
773        GO TO 72
      C    **********************************
      C    * PRINT LARGE WORDS SUB-SECTION *
      C    **********************************
      C TEST PROGRAM PARAMETER TO SEE IF LARGE WORDS ARE TO BE LISTED.
774     74 IF (SW1.EQ.2) GO TO 78
      C PRINT HEADING.
777        WRITE (6,75)
1000    75 FORMAT (1H0,15X,21HL A R G E   W O R D S)
1001       SINDX2=1
      C CHECK FOR AN EMPTY FILE.
1002       IF (SUPL2(1).NE.0) GO TO 76
1005       WRITE (6,71)
1006       GO TO 78
      C LOOK FOR THE -END OF FILE- INDICATOR.
1007    76 IF (SUPL2(SINDX2).EQ.0) GO TO 78
1012       NWRDS=SUPL2(SINDX2)
1013       NWINDX=SINDX2+NWRDS
1014       JSIN=SINDX2+1
      C PRINT THE LARGE WORD.
1015       WRITE (6,77) (SUPL2(J),J=JSIN,NWINDX)
1022    77 FORMAT (1H ,17X,14A6)
1023       SINDX2=NWINDX+1
1024       GO TO 76
      C IDENTIFY THE END OF THE ARTICLE
1025    78 WRITE (6,79)
1026    79 FORMAT (1H0,15X,27HE N D   O F   A R T I C L E)
      C GO CHECK FOR A NEW ARTICLE
1027       GO TO 2
1030    80 FORMAT (14A6)
1031       END
```

3. Dictionary Processing Program Listing

```
31571                  BAXEND                       FORTRAN SOURCE LIST
      ISN        SOURCE STATEMENT

        0 $IBFTC DICTDC
          C DICTIONARY PROCESSING AND CREATION OF TRANSFER MATRIX
        1       DIMENSION INDICT(12),DICT(3000),TRADD(4096)
        2       INTEGER DICT,TRADD,END,BLANK
        3       DATA END,BLANK/O777777777777,6H      /
        4       WRITE (6,71)
          C LINK IS A VALUE USED TO CHAIN DICT ENTRIES
          C NLINK IS USED TO GENERATE NEXT LINK VALUE
        5       NLINK=3
          C INDX REFERS TO INDEX OF TRANSFER MATRIX
        6       INDX=0
          C READ IN ONE LOGICAL RECORD - ONE DICT ENTRY
        7     1 READ (13,2) INDICT,NPACK
       12     2 FORMAT (12A6,4X,I4)
       13       LINK=NLINK
          C CHECK FOR END OF DICTIONARY
       14       IF (INDICT(1).EQ.END) GO TO 10
          C NWRD REFERS TO NUMBER OF MACHINE WORDS USED BY DICT ENTRY
       17       DO 3 NWRD =1,12
       20       IF (INDICT(NWRD).EQ.BLANK) GO TO 5
       23       DICT(NLINK)=INDICT(NWRD)
       24     3 NLINK=NLINK+1
          C IF PROGRAM COMES TO THIS POINT, AN ERROR EXISTS
       26       WRITE(6,4)INDICT
       27     4 FORMAT (1H ,21HERROR 1, LARGE ENTRY.,14A6)
       30       REWIND 13
       31       CALL EXIT
       32     5 NLINK=NLINK+2
       33    30 DICT(LINK-1)=NPACK
       34       DICT(LINK-2)=NLINK
       35       WRITE (6,70) NLINK,NPACK,INDICT
       36    70 FORMAT (1H ,2I6,5X,12A6)
       37    71 FORMAT  (33H1  ADDR PACK NO   DICTIONARY TERM/)
       40    33 IF (INDICT(1).LT.0) GO TO 6
          C MXINDX IS THE MATRIX INDEX GENERATED BY A DICT ENTRY
       43       MXINDX=INDICT(1)/2**24
       44       GO TO 7
       45     6 MXINDX=2**11-INDICT(1)/2**24
       46     7 IF (MXINDX-INDX) 1,9,8
       47     8 INDX=INDX+1
       50       TRADD(INDX)=0
       51       GO TO 7
       52     9 INDX=INDX+1
       53       TRADD(INDX)=LINK
       54       GO TO 1
          C SET CHAIN WORD OF LAST ENTRY TO 0
       55    10 DICT(LINK-1)=0
       56       DICT(LINK)=INDICT(1)
          C AT THIS POINT THE DICTIONARY IS (LINK) WORDS LONG
       57       REWIND 13
       60       REWIND 2
       61       WRITE (2) TRADD
       62       WRITE (2) LINK,(DICT(I),I=1,LINK)
       67       WRITE (6,11) TRADD
       70    11 FORMAT (1H0,10I6)
```

```
  ISN       SOURCE STATEMENT

      C BEGIN PROCESSING PACKAGE CARDS.
   71         INTEGER PACKNO,REGNO,PACK1(100),PACK5(350),PINDEX
   72         DIMENSION INPUT(12)
   73      40 READ (5,41) PACKNO,REGNO
   76      41 FORMAT (I6,I8)
   77         IF (PACKNO.EQ.999999) GO TO 67
  102         I=1+PACKNO/100
  103         GO TO (42,42,56), I
      C ****  4-WORD PACKAGE PROCESSING  ****
      C COMPUTE THE INDEX FOR THE 4-WORD PACKAGE
  104      42 PINDEX=PACKNO*4-3
  105         PACK5(PINDEX)=REGNO
      C READ AND PROCESS 3 DATA CARDS
  106         DO 52 I=1,3
  107         PINDEX=PINDEX+1
  110         READ (5,43) INPUT
  112      43 FORMAT (12A6)
  113         IF (INPUT(1).EQ.BLANK) GO TO 52
  116         DO 44 NWRDS =1,12
  117         IF (INPUT(NWRDS).EQ.BLANK) GO TO 45
  122      44 CONTINUE
  124      45 IF (INPUT(1).LT.0) GO TO 46
      C INDEX REFERS TO INDEX OF TRANSFER ADDRESS MATRIX
  127         INDEX=1+INPUT(1)/2**24
  130         GO TO 47
  131      46 INDEX=1+2**11-INPUT(1)/2**24
      C USE M AS INDEX OF PROCESSED DICTIONARY
  132      47 M=TRADD(INDEX)
  133         NWRDS=NWRDS-1
      C TEST THE NEXT 35 DICTIONARY WORDS FOR A COMPARISON.
  134         DO 49 K=1,35
      C TEST FOR THE END OF THE DICTIONARY
  135         IF (M.EQ.0) GO TO 54
  140         MTEMP=M
  141         DO 48 L=1,NWRDS
  142         IF (INPUT(L).NE.DICT(MTEMP)) GO TO 53
  145      48 MTEMP=MTEMP+1
      C NEXT DICT WORD SHOULD BE SOME INDEX LESS THAN 5000. IF THERE IS MORE
      C   LEFT TO THE WORD, THEN THIS TEST WILL FAIL MEANING WE MUST CONTINUE
      C   SEARCHING THE DICTIONARY FOR A COMPARISON.
  147         IF (IABS(DICT(MTEMP)).LT.5000)  GO TO 51
  152      53 M=DICT(M-2)
  153      49 CONTINUE
      C AT THIS POINT WE HAVE CHECKED 35 WORDS BUT COULD FIND NO COMPARISON.
  155         WRITE (6,50) PACKNO,INPUT
  156      50 FORMAT (12H0RE PACK NO.,I4,13H CAN NOT FIND/1H ,5X,12A6,
             113HIN DICTIONARY)
  157         GO TO 52
      C INSERT DICTIONARY INDEX IN PACKAGE
  160      51 PACK5(PINDEX)=M
  161      52 CONTINUE
      C GO READ IN NEXT PACKAGE DATA CARD
  163         GO TO 40
  164      54  WRITE (6,55) PACKNO,INPUT
  165      55 FORMAT (12H0RE PACK NO.,I4,37H REACHED END OF DICTIONARY PROCESSIN
31571                 BAXEND                   FORTRAN SOURCE LIST DICTCD
      ISN        SOURCE STATEMENT

               10/1H ,5X,12A6)
      166          GO TO 52
             C****  1-WORD PACKAGE PROCESSING  ****
      167      56 PINDEX=PACKNO-200
      170         READ (5,43) INPUT
      172         DO 58 NWRDS=1,12
      173         IF (INPUT(NWRDS).EQ.BLANK) GO TO 59
      176      58 CONTINUE
      200      59 NWRDS=NWRDS-1
      201         IF (INPUT(1).LT.0) GO TO 60
      204         INDEX=1+INPUT(1)/2**24
      205         GO TO 61
      206      60 INDEX=1+2**11-INPUT(1)/2**24
      207      61 M=TRADD(INDEX)
      210         DO 63 K=1,35
      211         IF (M.EQ.0) GO TO 66
      214         MTEMP=M
      215         DO 62 L=1,NWRDS
      216         IF (INPUT(L).NE.DICT(MTEMP)) GO TO 65
      221      62 MTEMP=MTEMP+1
      223         IF (IABS(DICT(MTEMP)).LT.5000) GO TO 64
      226      65 M=DICT(M-2)
      227      63 CONTINUE
      231         WRITE (6,50) PACKNO,INPUT
      232         GO TO 40
      233      64 PACK1(PINDEX)=M
      234         GO TO 40
      235      66 WRITE (6,55) PACKNO,INPUT
      236         GO TO 40
      237      67 WRITE (2) PACK5,PACK1
      240         WRITE (6,72)
      241         WRITE (6,68) (PACK5(I),I=1,236)
      246      68 FORMAT (1H0,I9,3I5)
      247         WRITE (6,73)
      250         WRITE (6,69) (I,PACK1(I),I=1,15)
      255      69 FORMAT (1H0,I9,I5)
      256         REWIND 2
      257         CALL EXIT
      260      72 FORMAT (22H1CODED TYPE 1 PACKAGES/)
      261      73 FORMAT (22H1CODED TYPE 2 PACKAGES/)
      262         END
```

4. Sorted Dictionary Terms with Linkages and Package Numbers

```
ACCR PACK NO   DICTIONARY TERM

  11    -5     1-AMINOPROPANE-1,3-DICARBOXYLIC ACID
  20   -19     1-ETHYL-2,5-DIOXO-4-PHENYLIMIDAZOLIDINE
  26    -4     2-AMINOGLUTAMARIC ACID
  32    -5     2-AMINOPENTANEDIOIC ACID
  38   -23     2-DESOXYPHENOBARBITAL
  45    17     2-ETHYL,2-METHYLSUCCINIMIDE
  52   -38     2-HYDROXY-2-PHENYLPROPIONAMIDE
  62    10     2-METHYL-2-N-PROPYL-1,3-PROPANEDIOL DICARBAMATE
  72    10     2-METHYL-2-PROPYL-1,3-PROPANEDIOL DICARBAMATE
  79   -38     2-PHENYL-2-HYDROXYPROPIONAMIDE
  88   -25     2-(P-AMINOPHENYL)-2-ETHYL-GLUTARIMIDE
  96    10     2,2-DI(CARBAMOYLOXYMETHYL)PENTANE
 104    37     3-ALLYL-5-ISOBUTYL-2-THIOHYDANTOIN
 113    34     3-ALLYL-5-METHYL-2,4-OXAZOLIDINEDIONE
 122   -34     3-ALLYL-5-METHYLOXAZOLIDINE-2,4-DIONE
 129    19     3-ETHYL-5-PHENYLHYDANTOIN
 139   -33     3-ETHYL-5,5-DIMETHYL-2,4-DIKETO-OXAZOLIDINE
 146   -57     3-METHYL-5-PHENYLHYDANTOIN
 154    -3     3-METHYL-5-PHENYL-5-ETHYL HYDANTOIN
 162    -3     3-METHYL-5,5-PHENYLETHYLHYDANTOIN
 171    22     3,5-DIMETHYL-5-ETHYLOXAZOLIDINE-2,4-DIONE
 179   -27     3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE
 187   -27     3,5,5-TRIMETHYLOXAZOLIDINE-2,4-DIONE
 195    -3     3,METHYL-5,5-PHENYLETHYLHYDANTOIN
 206   -23     5-ETHYLDIHYDRO-5-PHENYL-4,6(1H,5H)-PYRIMIDINEDIONE
 215    21     5-ETHYL-1-METHYL-5-PHENYLBARBITURIC ACID
 223    41     5-ETHYL-1-METHYL-5-PHENYLHYDANTOIN
 231     3     5-ETHYL-3-METHYL-5-PHENYLHYDANTOIN
 242    -3     5-ETHYL-3-METHYL-5-PHENYL-2,4(3H,5H)-IMIDAZOLEDIONE
 251   -22     5-ETHYL-3,5-DIMETHYL-2,4-OXAZOLIDINEDIONE
 259    -9     5-ETHYL-5-ISOAMYLBARBITURIC ACID
 267    -9     5-ETHYL-5-ISOPENTYLBARBITURIC ACID
 275     1     5-ETHYL-5-PHENYLBARBITURIC ACID
 285   -23     5-ETHYL-5-PHENYLHEXAHYDROPYRIMIDINE-4,6-DIONE
 294   -21     5-ETHYL-5-PHENYL-N-METHYL-BARBITURIC ACID
 302    15     5-METHYL-5-PHENYLBARBITURIC ACID
 312    23     5-PHENYL-5-ETHYLHEXAHYDROPYRIMIDINE-4,6-DIONE
 321   -21     5-PHENYL-5-ETHYL-3-METHYLBARBITURIC ACID
 329    24     5-PHENYL-5-(2-THIENYL)-HYDANTOINATE
 337    39     5-(31-PHENANTHRYL)-5-METHYLHYDANTOIN
 345     2     5,5-DIETHYL-1-METHYLBARBITURIC ACID
 354    33     5,5-DIMETHYL-3-ETHYL-OXAZOLIDINEDIONE
 362    36     5,5-DIMETHYLOXAZOLIDINE-2,4-DIONE
 368    -8     5,5-DIPHENYLHYDANTOIN
 375   -35     5,5-DIPHENYLHYDANTOIN SODIUM
 383     8     5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE
 392   -35     5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE SODIUM
 401   -35     5,5-DIPHENYL-2,4-OXAZOLIDINEDIONE SODIUM
 409   -35     5,5-DIPHENYL-HYDANTOINATE SODIUM
 416   -40     5,5-DIPHENYL-OXAZOLIDINEDIONE
 423    40     5,5-DIPROPYL-OXAZOLIDINEDIONE
 431   -29     6-ETHOXYBENZOTHIAZOLE-2-SULFONAMIDE
 435  -210      ABSORPTION
 438  -201      ACTING
 441  -201      ACTION
 444  -201      ACTIVE
 448  -201      ACTIVITY
```

```
452  -204    ADMINISTERED
456  -204    ADMINISTER
461  -204    ADMINISTRATION
466  -208    ADVERSE EFFECT
471  -208    ADVERSE EFFECTS
476  -208    ADVERSE REACTION
481  -208    ADVERSE REACTIONS
485     1    AGRYPNAL
489    37    ALBUTOIN
493    44    ALEPSAL
497    35    ALEPSIN
501    34    ALOXIDONE
507    -5    ALPHA-AMINOGLUTARIC ACID
515   -17    ALPHA-ETHYL-ALPHA-METHYLSUCCINIMIDE
524   -38    ALPHA-HYDROXY-ALPHA-PHENYLPROPIONAMIDE
532   -20    ALPHA-PHENYL-ALPHA-ETHYLACETYLUREA
542    25    ALPHA(P-AMINOPHENYL)-ALPHA ETHYL-GLUTARIMIDE
547   -25    AMINOGLUTETHIMIDE
552    25    AMINO-GLUTETHIMIDE
556    -9    AMOBARBITAL
561   -14    AMOBARBITAL SODIUM
566    -9    AMYLOBARBITONE
569     9    AMYTAL
574    14    AMYTAL SODIUM
579   -49    ANTICONVULSANT
584   -49    ANTICONVULSANTS
589   -49    ANTI-CONVULSANT
594  -203    ANTI-EPILEPTIC
598    35    ANTISACER
603    59    ANTISPASMODIC
607    10    APASCIL
611    10    APPETROL
615    10    ARCOBAN
619    10    ATRAXIN
624    38    ATROLACTAMIDE
628    39    BAGROSIN
632     9    BARBAMIL
636     9    BARBAMYL
640     1    BARBENYL
644     1    BARBIPHENYL
648     1    BARBIPIL
652     1    BARBITA
656   -50    BARBITURATE
660   -50    BARBITURATES
664     1    BARBIVIS
668     7    BARPENTAL
672    30    BECLAMIDE
677   -30    BENZCHLORPROPAMIDE
681    10    BIOBAMAT
685     1    BLU-PHEN
689   -56    BULL NETTLE
693    10    CALMIREN
697    10    CANQUIL
702    28    CARBAMAZEPINE
706    29    CARDRASE
710    16    CELONTIN
714    30    CHLORACON
720    30    CHLOROETHYLPHENAMIDE
723    10    CIRPON
```

727	35	CITRULLAMON
731	46	COMITAL
735	12	CONTRAVUL
738	10	CYRPON
742	11	DELACURANINE
746	41	DELTOIN
750	35	DENYL SODIUM
754	10	DESA-BAMATE
758	47	DESBUTAL
762	8	DIHYCON
767	35	DIHYDAN SOLUBLE
770	8	DI-LAN
774	8	DILANTIN
779	35	DILANTIN SODIUM
783	33	DIMEDIONE
787	36	DIMETHADIONE
791	35	DIPHANTOINE
796	35	DIPHENINE SODIUM
800	35	DIPHENTOIN
805	35	DIPHENYLAN SODIUM
811	-35	DIPHENYLHYDANTOINATE
816	8	DIPHENYLHYDANTOIN
822	35	DIPHENYLHYDANTOIN SODIUM
826	31	DIPHOXAZIDE
832	-11	D-TUBOCURARINE CHLORIDE
836	14	DORMINAL
840	1	DORMIRAL
844	9	DORMYTAL
847	-205	DOSAGE
850	-205	DOSE
853	-205	DOSES
857	1	DUNERYL
861	10	ECUANIL
866	-206	EFFECTIVENESS
870	-206	EFFECTIVE
873	-206	EFFECT
877	-206	EFFECTS
881	-206	EFFICACY
884	8	EKKO
888	25	ELIPTEN
892	7	EMBUTAL
896	35	EPANUTIN
899	8	EPELIN
903	13	EPICLASE
907	27	EPIDIONE
911	40	EPIDONE
915	-207	EPILEPSY
919	-207	EPILEPTIC
923	-55	EPSOM SALT
926	35	EPTOIN
930	10	EQUANIL
934	1	ESCABARB
938	1	ESKABARB
942	17	ETHOSUXIMIDE
946	19	ETHOTOIN
951	-29	ETHOXZOLAMIDE
955	1	ETILFEN
959	1	EUNERYL
963	36	EUPRACTONE

966	37	EUPRAX
970	1	GARDENAL
975	6	GARDENAL SODIUM
979	1	GARDEPANYL
982	35	GAROIN
986	2	GEMONIL
990	5	GLUTACID
996	-4	GLUTAMIC ACIC 5-AMIDE
1001	-5	GLUTAMIC ACID
1005	5	GLUTAMINE
1010	-5	GLUTAMINIC ACID
1014	10	HARMONIN
1018	58	HARMONYL-N
1022	30	HIBICON
1026	35	HIDANTAL
1030	-56	HORSE NETTLE
1034	43	HYDANTAL
1038	-51	HYDANTOINS
1042	1	HYPNETTES
1046	3	INSULTON
1051	-215	INVESTIGATION
1058	-9	ISOAMYL-ETHYLBARBITURIC ACID
1062	7	ISOBARB
1066	9	ISOMYTAL
1069	21	ISONAL
1074	35	LEPITOIN SODIUM
1077	18	LIFENE
1081	1	LIQUITAL
1085	1	LIXOPHEN
1089	1	LUMINAL
1094	6	LUMINAL SODIUM
1099	-55	MAGNESIUM SULFATE
1103	34	MALAZOL
1107	34	MALICONE
1111	21	MEBARAL
1115	46	MEBROIN
1119	48	MEDYLAN
1123	10	MEPANTIN
1127	10	MEPAVLON
1131	3	MEPHENYTOIN
1136	21	MEPHOBARBITAL
1140	10	MEPROBAMATE
1144	10	MEPROCON
1148	10	MEPROSIN
1152	10	MEPROSPAN
1156	10	MEPROTABS
1160	3	MESANTOIN
1164	3	MESONTOIN
1168	2	METHARBITAL
1172	41	METHETOIN
1176	3	METHOIN
1180	16	METHSUXIMIDE
1186	-3	METHYLPHENYLHYDANTOIN
1192	-18	METHYLPHENYLSUCCINIMIDE
1196	18	MILONTIN
1200	10	MILTOWN
1204	35	MINETOIN
1207	38	M-144
1217	-7	MONOSODIUM 5-ETHYL-5-(1-METHYLBUTYL)BARBITURATE

1226	-7	MONOSODIUM ETHYL(1-METHYLBUTYL)BARBITURATE
1230	23	MYLEPSIN
1234	23	MYSOLINE
1247	31	N1-(BETA-HYCROXY-BETA-BETA-CIPHENYLPROPIONYL)-N2-ACETYLHYDRAZIN
1251	7	NA-PENT
1255	7	NAPENTAL
1259	58	NEMBU-DONNA
1263	58	NEMBU-GESIC
1267	7	NEMBUTAL
1271	10	NERVONUS
1275	1	NEUROBARB
1281	-211	NEUROPHARMACOLOGICAL
1286	-211	NEUROPHARMACOLOGY
1294	-30	N-BENZYL-BETA-CHLOROPROPANAMIDE
1302	30	N-BENZYL-BETA-CHLOROPROPIONAMIDE
1310	-21	N-METHYLETHYLPHENYLBARBITURIC ACID
1317	18	N-METHYL-2-PHENYLSUCCINIMIDE
1325	-3	N-METHYL-5,5-PHENYLETHYLHYDANTOIN
1335	-16	N-METHYL-ALPHA-METHYL-ALPHA-PHENYLSUCCINIMIDE
1343	-18	N-METHYL-ALPHA-PHENYLSUCCINIMIDE
1353	-16	N-METHYL-ALPHA,ALPHA-METHYLPHENYLSUCCINIMIDE
1359	-21	N-METHYL PHENOBARBITAL
1366	-20	N-(ALPHA-PHENYLBUTYRYL)UREA
1369	1	NOPTIL
1372	1	NUNOL
1376	57	NUVARONE
1380	30	NYDRANE
1388	-16	N,2-DIMETHYL-2-PHENYLSUCCINIMIDE
1391	10	OASIL
1395	7	OMNISED
1399	12	OSPOLOT
1404	-52	OXAZOLIDINEDIONES
1408	10	PANADIOL
1412	22	PARADIONE
1417	22	PARAMETHADIONE
1421	19	PEGANONE
1424	7	PENTAL
1430	7	PENTOBARBITAL SODIUM
1436	-7	PENTOBARBITONE SODIUM
1440	7	PENTONE
1443	7	PENTYL
1447	-9	PENTYMAL
1451	10	PEREQUIL
1455	10	PERQUIETIL
1459	10	PERTRANQUIL
1463	27	PETIDIONE
1467	27	PETIDION
1471	-27	PETIDON
1475	13	PHACETUR
1480	-211	PHARMACOLOGICAL
1484	-211	PHARMACOLOGY
1488	45	PHELANTIN
1492	-21	PHEMITONE
1496	13	PHENACEMIDE
1502	-13	PHENACETYL-CARBAMIDE
1506	3	PHENANTOIN
1510	-20	PHENETURIDE
1515	35	PHENITOIN SODIUM
1519	1	PHENOBAL

1524	1	PHENOBARBITAL
1530	6	PHENOBARBITAL SODIUM
1535	-1	PHENOBARBITONE
1539	1	PHENONYL
1543	1	PHENOTURIC
1547	18	PHENSUXIMIDE
1551	13	PHENURONE
1556	-13	PHENYLACETYLUREA
1561	-13	PHENYLACETYLUREA
1568	1	PHENYLETHYLBARBITURIC ACID
1574	-1	PHENYLETHYLMALONYLUREA
1581	-15	PHENYLMETHYLBARBITURIC ACID
1585	-8	PHENYTOIN
1590	-35	PHENYTOIN SODIUM
1595	35	PHENYTOIN SOLUBLE
1599	24	PHETHENYLATE
1605	42	PHETHENYLATE SODIUM
1608	1	PHOB
1621	-12	P-(TETRAHYDRO-2H-1,2-THIAZIN-2-YL)-BENZENESULFONAMIDE,S,S,DIOXIDE
1625	10	PLACIDON
1629	30	POSEDRINE
1633	23	PRIMIDONE
1637	10	PROBATE
1641	10	PROMABYL
1645	10	PROMATE
1649	21	PROMINAL
1653	32	PROPAZONE
1657	-213	PROPERTIES
1661	-213	PROPERTY
1664	27	PTIMAL
1669	-53	PYRIMIDINEDIONES
1672	10	QUANIL
1676	-56	RADICAL WEED
1680	15	RUTONAL
1684	3	SACERNO
1688	7	SAGATAL
1692	-56	SAND-BRIER
1696	3	SEDANTOIN
1700	10	SEDAZIL
1704	-208	SIDE EFFECT
1708	-208	SIDE EFFECTS
1713	35	SILANTIN SODIUM
1721	-14	SODIUM 5-ETHYL-5-ISOAMYLBARBITURATE
1729	6	SODIUM 5-ETHYL-5-PHENYLBARBITURATE
1739	7	SODIUM 5-ETHYL-5-(1-METHYLBUTYL)BARBITURATE
1748	42	SODIUM 5-PHENYL-5(2-THIENYL)HYDANTOINATE
1757	-35	SODIUM 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE
1765	35	SODIUM 5,5-DIPHENYL HYDANTOINATE
1772	-14	SODIUM ISOAMYLETHYLBARBITURATE
1780	-6	SODIUM PHENYL-5-ETHYL-5-BARBITURATE
1784	-26	SOLANINE
1788	35	SOLANTOIN
1792	35	SOLANTYL
1796	-56	SOLANUM
1800	-26	SOLATUNINE
1806	-7	SOLUBLE PENTOBARBITAL
1812	-6	SOLUBLE PHENOBARBITAL
1818	-6	SOLUBLE PHENOBARBITONE
1821	9	SOMNAL

1825	1	SOMONAL
1829	7	SOPENTAL
1832	7	SOTYL
1836	1	STENTAL
1840	-54	SUCCINIMIDES
1844	12	SULPHENYTAME
1848	12	SULTHIAME
1852	35	TACOSAL
1856	28	TEGRETOL
1860	1	TEOLAXIN
1872	12	TETRAHYDRO-2-P-SULFAMOYL-PHENYL-1,2-THIAZINE-1,1-DIOXIDE
1876	38	THEMISONE
1881	-214	THERAPEUTIC EFFECT
1885	-214	THERAPEUTIC
1889	-214	THERAPY
1893	-24	THIANTOINE
1898	-42	THIANTOINE SODIUM
1902	24	THIANTOIN
1907	42	THIANTOIN SODIUM
1911	-24	THYPHENTOIN
1915	-212	TOXICITY
1918	-212	TOXIC
1922	-214	TREATED
1926	-214	TREATMENT
1929	-214	TREAT
1933	27	TRIDIONE
1937	27	TRIMEDAL
1942	27	TRIMETHADIONE
1946	1	TRIPHENATOL
1950	12	TROLONE
1954	27	TROXIDONE
1958	11	TUBADIL
1962	11	TUBARINE
1968	11	TUBOCURARINE CHLORIDE
1971	10	URBIL
1975	10	VIO-BAMATE
1979	17	ZARONTIN
1983	35	ZENTROPIL
1989	-20	(2-PHENYLBUTYRYL)UREA

5. Type 1 and Type 2 Packages -- Input

```
     1   50066
5-ETHYL-5-PHENYLBARBITURIC ACID
PHENOBARBITAL
BARBITURATES
     2   50113
GEMONIL
5,5-DIETHYL-1-METHYLBARBITURIC ACID
BARBITURATES
     3   50124
MEPHENYTOIN
5-ETHYL-3-METHYL-5-PHENYLHYDANTOIN
HYDANTOINS
     4   56859
2-AMINOGLUTAMARIC ACID


     5   56860
GLUTAMIC ACID
1-AMINOPROPANE-1,3-DICARBOXYLIC ACID

     6   57307
PHENOBARBITAL SODIUM
SODIUM 5-ETHYL-5-PHENYLBARBITURATE
BARBITURATES
     7   57330
PENTOBARBITAL SODIUM
SODIUM 5-ETHYL-5-(1-METHYLBUTYL)BARBITURATE
BARBITURATES
     8   57410
DIPHENYLHYDANTOIN
5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE
HYDANTOINS
     9   57432
AMOBARBITAL
5-ETHYL-5-ISOAMYLBARBITURIC ACID
BARBITURATES
    10   57534
MEPROBAMATE
2-METHYL-2-PROPYL-1,3-PROPANEDIOL DICARBAMATE

    11   57943
TUBOCURARINE CHLORIDE


    12   61563
TETRAHYDRO-2-P-SULFAMOYL-PHENYL-1,2-THIAZINE-1,1-DIOXIDE


    13   63989
PHENACEMIDE
PHENYLACETYLUREA

    14   64437
AMOBARBITAL SODIUM
SODIUM 5-ETHYL-5-ISOAMYLBARBITURATE
BARBITURATES
    15   76948
5-METHYL-5-PHENYLBARBITURIC ACID
BARBITURATES
```

```
    16   77418
CELONTIN
N,2-DIMETHYL-2-PHENYLSUCCINIMIDE
SUCCINIMIDES
    17   77678
ETHOSUXIMIDE
2-ETHYL,2-METHYLSUCCINIMIDE
SUCCINIMIDES
    18   86340
PHENSUXIMIDE
N-METHYL-2-PHENYLSUCCINIMIDE
SUCCINIMIDES
    19   86351
ETHOTOIN
3-ETHYL-5-PHENYLHYDANTOIN
HYDANTOINS
    20   90493
(2-PHENYLBUTYRYL)UREA
PHENETURIDE

    21  115388
MEPHOBARBITAL
5-ETHYL-1-METHYL-5-PHENYLBARBITURIC ACID
BARBITURATES
    22  115673
PARAMETHADIONE
3,5-DIMETHYL-5-ETHYLOXAZOLIDINE-2,4-DIONE
OXAZOLIDINEDIONES
    23  125337
PRIMIDONE
5-PHENYL-5-ETHYLHEXAHYDROPYRIMIDINE-4,6-DIONE
BARBITURATES
    24  125611
PHETHENYLATE
5-PHENYL-5-(2-THIENYL)-HYDANTOINATE
HYDANTOINS
    25  125848
AMINOGLUTETHIMIDE
ALPHA(P-AMINOPHENYL)-ALPHA ETHYL-GLUTARIMIDE

    26  125973
SOLANINE


    27  127480
TRIMETHADIONE
3,5,5-TRIMETHYL-2,4-OXAZOLIDINEDIONE
OXAZOLIDINEDIONES
    28  298464
CARBAMAZEPINE
TEGRETOL

    29  452357
ETHOXZOLAMIDE
6-ETHOXYBENZOTHIAZOLE-2-SULFONAMIDE

    30  501688
N-BENZYL-BETA-CHLOROPROPIONAMIDE
CHLOROETHYLPHENAMIDE

    31  511411
```

```
N1-(BETA-HYDROXY-BETA-BETA-DIPHENYLPROPIONYL)-N2-ACETYLHYDRAZINE


    32  512129
5,5-DIPROPYL-OXAZOLIDINEDIONE


    33  520774
5,5-DIMETHYL-3-ETHYL-OXAZOLIDINEDIONE
OXAZOLIDINEDIONES

    34  526352
ALOXIDONE
3-ALLYL-5-METHYL-2,4-OXAZOLIDINEDIONE
OXAZOLIDINEDIONES
    35  630933
DIPHENYLHYDANTOIN SODIUM
SODIUM 5,5-DIPHENYL HYDANTOINATE
HYDANTOINS
    36  695534
DIMETHADIONE
5,5-DIMETHYLOXAZOLIDINE-2,4-DIONE
OXAZOLIDINEDIONES
    37  830897
ALBUTOIN
3-ALLYL-5-ISOBUTYL-2-THIOHYDANTOIN
HYDANTOINS
    38 2019683
2-HYDROXY-2-PHENYLPROPIONAMIDE


    39 3784927
5-(31-PHENANTHRYL)-5-METHYLHYDANTOIN
HYDANTOINS

    40 4171113
5,5-DIPHENYL-OXAZOLIDINEDIONE
OXAZOLIDINEDIONES

    41 5696060
METHETOIN
5-ETHYL-1-METHYL-5-PHENYLHYDANTOIN
HYDANTOINS
    42 6509348
PHETHENYLATE SODIUM
SODIUM 5-PHENYL-5(2-THIENYL)HYDANTOINATE
HYDANTOINS
    43-8028679
HYDANTOINS
BARBITURATES

    44-8028680



    45-8028691
HYDANTOINS
BARBITURATES

    46-8028704
HYDANTOINS
```

BARBITURATES

47-8028715
BARBITURATES

48-8028726
HYDANTOINS
BARBITURATES

49
ANTICONVULSANTS

50
BARBITURATES

51
HYDANTOINS

52
OXAZOLIDINEDIONES

53
PYRIMIDINEDIONES

54
SUCCINIMIDES

55
MAGNESIUM SULFATE
EPSOM SALT

56
BULL NETTLE
SOLANUM

57
3-METHYL-5-PHENYLHYDANTOIN
HYDANTOINS

58
BARBITURATES

59
ANTISPASMODIC

```
   201
ACTIVITY
   203
ANTI-EPILEPTIC
   204
ADMINISTRATION
   205
DOSAGE
   206
EFFECT
   207
EPILEPSY
   208
ADVERSE REACTIONS
   210
ABSORPTION
   211
PHARMACOLOGY
   212
TOXICITY
   213
PROPERTIES
   214
THERAPY
   215
INVESTIGATION
```

6. Coded Type 1 and Type 2 Packages

```
CODED TYPE 1 PACKAGES

     50066  267 1519  656

     50113  982  337  656

     50124 1127  223 1034

     56859   20    0    0

     56860  996    3    0

     57307 1524 1721  656

     57330 1424 1729  656

     57410  811  375 1034

     57432  552  251  656

     57534 1136   62    0

     57943 1962    0    0

     61563 1860    0    0

     63989 1492 1551    0

     64437  556 1713  656

     76948  294  656    0

     77418  706 1380 1836

     77678  938   38 1836

     86340 1543 1310 1836

     86351  942  122 1034

     90493 1983 1506    0

    115388 1131  206  656

    115673 1412  162 1399

    125337 1629  302  656

    125611 1595  321 1034

    125848  542  532    0

    125973 1780    0    0

    127480 1937  171 1399

    298464  697 1852    0

    452357  946  423    0
```

```
  501688 1294  714    0
  511411 1234    0    0
  512129  416    0    0
  520774  345 1399    0
  526352  497  104 1399
  630933  816 1757 1034
  695534  783  354 1399
  830897  485   96 1034
 2019683   45    0    0
 3784927  329 1034    0
 4171113  409 1399    0
 5696060 1168  215 1034
 6509348 1599 1739 1034
-8028679 1034  656    0
-8028680    0    0    0
-8028691 1034  656    0
-8028704 1034  656    0
-8028715  656    0    0
-8028726 1034  656    0
      -0  579    0    0
      -0  656    0    0
      -0 1034    0    0
      -0 1399    0    0
      -0 1664    0    0
      -0 1836    0    0
      -0 1094  919    0
      -0  685 1792    0
      -0  139 1034    0
      -0  656    0    0
      -0  598    0    0
```

```
CODED TYPE 2 PACKAGES

          1  444

          2    0

          3  589

          4  456

          5  844

          6  870

          7  911

          8  476

          9    0

         10  431

         11 1480

         12 1911

         13 1653

         14 1885

         15 1046
```

7. Sample of Print-out of Transfer Address Matrix

```
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0  1069     0     0     0  1074     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0  1085     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0  1094     0     0     0  1107     0     0     0  1192
     0     0     0     0     0     0  1204     0     0     0
     0     0  1207     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
  1226     0     0     0     0     0     0     0     0  1234
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0  1247     0     0     0  1255
     0     0     0     0     0     0     0     0     0     0
  1286     0     0     0     0     0  1366     0     0     0
     0     0     0     0     0     0     0     0     0     0
  1369     0     0     0  1376     0     0  1380     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0  1388
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0  1391     0
     0     0     0     0     0     0     0     0     0     0
     0     0  1395     0     0     0     0  1399     0     0
     0     0     0     0     0     0     0     0     0     0
```

8. Subroutine Program Listings

```
      $TEXT   COMPOS                                                        COMPOO

      * THIS SUBROUTINE TAKES SINGLE CHARACTERS (A1 FORMAT) AND PACKS THEM
      * SIX TO A WORD (A6 FORMAT).  THE FOLLOWING VARIABLES ARE USED IN
      * UNLABELLED COMMON-
      *         - ARRAY WHERE PACKED WORDS ARE RETURNED (14 WORDS).
      *         - INTEGER VALUE SHOWING NUMBER OF PACKED WORDS RETURNED.
      *         - ARRAY OF INDIVIVIDUAL CHARACTERS (72 WORDS).
      *         - INTEGER VALUE SHOWING NUMBER OF CHARACTERS TO BE PROCESSED.
                ENTRY   COMPOS
                CONTRL  //
      COMPOS    SAVE    1,2,4
 10011          LXA     CHAR,1           PICK UP THE NUMBER OF CHARACTERS TO BE DONE
 10000          AXT     72,4             PROCESS A MAXIMUM OF 72.
 10000          ZAC                      CLEAR THE AC.
 10011 LOOP     STO     WORDS            STORE COUNT OF NUMBER OF PACKED WORDS.


 10001          ADD     INIT             FIND ADDRESS WHERE NEXT PACKED WORD GOES.
 10001          STO     IA               SAVE ADDRESS.
 10000          AXT     6,2              NOW PROCESS 6 CHARACTERS.
 10011          LDQ     A1+72,4          PICK UP NEXT CHARACTER.
 10010
 10011
 10000          LGL     6                SHIFT INTO AC.
 10001          TXI     *+1,4,-1         DECREASE COUNT.
 10001          TIX     *+2,1,1          TEST TO SEE IF ALL CHARACTERS DONE.
 10001          TRA     DONE             YES, GO TO END OF SUBROUTINE.
 10001          TIX     *-5,2,1          GO BACK IF WE DONT HAVE 6 CHARACTERS PACKED
 10001          SLW*    IA               STORE PACKED WORD IN ADDRESS CALCULATED.
 10001          CLA     =1               INCREASE COUNT OF WORDS COMPLETED.
 10011          ADD     WORDS
 10001          TRA     LOOP             CONTINUE LOOP.
 10000          LGL     6                SHIFT BLANK CHARACTER INTO AC.
 10001 DONE     LDQ     BLANK            PICK UP BLANK CHARACTER TO FILL IN WORD.
 10001          TIX     *-2,2,1          SEE IF WORD HAS ALL 6 CHARACTERS.
 10001          SLW*    IA               STORE LAST WORD.


 10011          CLA     WORDS            INCREASE COUNT OF WORDS DONE
 10001          ADD     =1                 BY ONE.
 10011          STO     WORDS            STORE COUNT.
                RETURN  COMPOS           RETURN TO CALLING PROGRAM.
 00001 IA       BSS     1
 10011 INIT     PZE     A6
 10000 BLANK    BCI     1,
 00001          USE     //
 00001 A6       BSS     14
 00001 WORDS    BSS     1
 00001 A1       BSS     72
 00001 CHAR     BSS     1
 00001          USE     PREVIOUS
                    LITERALS
 10000
```

```
                EXTERN     S.SLOC   GENERATED
1111            END

COMPOS
```

```
                 BAXENDA                         FORTRAN SOURCE LIST
ISN          SOURCE STATEMENT

  0 $IBFTC DECOMP  NODECK
    C           SUBROUTINE TO DECOMPOSE A GIVEN NUMBER OF COMPUTER WORDS (A6)
    C           INTO CHARACTERS(A1)
  1        SUBROUTINE DECOMP
  2        DIMENSION CHAR(6),KHAR(6),MASK(10),AMSK(10)
  3        DATA MASK   /0770000000000,07700000000,077000000,0770000,07700,
          1077,0200000000000,0177777777777,0400000000000,0006060606060/
  4        EQUIVALENCE (CHAR(1),KHAR(1)),(MASK(1),AMSK(1))
  5        COMMON WRD(14),NWRDS,CHR(72),LENGTH
  6        DATA BLANK/1H /
    C           DECOMPOSE (NWRDS) 7040 WORDS INTO AN ARRAY OF (LENGTH) CHARACTERS
  7        DO 8 K = 1,NWRDS
 10        WORD = WRD(K)
    C           DECOMPOSE A 7040 WORD INTO AN ARRAY OF SIX CHARACTERS
 11        CHAR(1) = AND(WORD,AMSK(1))
 12        DO 6  I = 2,6
 13        J = I - 1
 14        CHAR(I) = AND(WORD, AMSK(I))
 15        KHAR(I) = KHAR(I)*2**(J*6 - 1)
    C           TEST FOR PRESENCE OF SIGN IN MOST SIGNIFICANT POSITION
 16        TEST = AND(CHAR(I), AMSK(7))
 17        IF (TEST. EQ. AMSK(7))  GO TO 5
 22        KHAR(I) = KHAR(I)*2
 23        GO TO 6
    C           REMOVE LEFTMOST BIT SHIFT AND REPLACE
 24      5 CHAR(I) = AND(CHAR(I), AMSK(8))
 25        KHAR(I) = KHAR(I)*2
 26        CHAR(I) = OR(CHAR(I), AMSK(9))
 27      6 CONTINUE
 31        DO 8 I = 1,6
 32        CHAR(I) = OR(CHAR(I),AMSK(10))
 33        IN = (K-1)*6 + I
 34      8 CHR(IN) = CHAR(I)
 37      9 IF(CHR(IN).NE.BLANK) GO TO 10
 42        IN = IN - 1
 43        GO TO 9
 44     10 LENGTH = IN
 45        RETURN
 46        END
```

```
                BAXENCA                        FORTRAN SOURCE LIST
ISN          SOURCE STATEMENT

  0 $IBFTC HEAD    NODECK
  1          SUBROUTINE HEAD
    C        THIS SUBROUTINE READS IN THE BIBLOGRAPHIC ENTRY CARDS OF EACH
    C        ARTICLE.  IT CONTINUES READING CARDS UNTIL A SIGNAL CARD IS
    C        ENCOUNTERED.  THE INFORMATION ON THESE CARDS ARE PRINTED WITH
    C        A MARGIN WIDTH THAT IS SPECIFIED IN THE LABELLED COMMON AREA -H-.
  2          COMMON /H/ NUM
  3          DATA EX,BLANK/1HX,1H /
  4          INTEGER EX,BLANK,RECORD,PRINT
  5          DIMENSION RECORD(73),PRINT(132)
  6          INTEGER V,Z,U,Y,W
  7          WRITE (6,1)
 10        1 FORMAT (1H1)
    C        SET UP OUTPUT PARAMETER J.
 11        2 J = 1
 12          W = 1
    C        SET UP INPUT PARAMETER M.
 13        3 M = 0
 14          READ (5,4)  (RECORD(Z), Z = 1,72)
 21        4 FORMAT(72A1)
 22          IF(.NOT.((RECORD(1).EQ.BLANK).AND.(RECORD(2).EQ.BLANK))) GO TO 6
 25          DO 5  Y = W,NUM
 26          PRINT(Y) = BLANK
 27        5 CONTINUE
 31          GO TO 12
    C        CHECK FOR THE SIGNAL CARD.
 32        6 IF (RECORD(1) .EQ. EX) GO TO 18
 35        7 DO 9  I = J,NUM
 36          M = M+1
 37          PRINT(I) = RECORD(M)
 40          IF (M .LT. 72) GO TO 8
 43          IF (I.EQ.NUM) GO TO 12
 46          GO TO 11
 47        8 IF ((RECORD(M).EQ.BLANK).AND.(RECORD(M+1).EQ.BLANK).AND.(RECORD(M+
          12).EQ.BLANK))  GO TO 12
 52        9 CONTINUE
    C        IF WORD IS INCOMPLETE AT END OF OUTPUT LINE, MOVE ENTIRE WORD
    C           TO NEXT LINE.
 54          L = NUM
 55       10 IF (PRINT(L) .EQ. BLANK) GO TO 13
 60          L = L-1
 61          GO TO 10
 62       11 J = I+1
 63          GO TO 3
 64       12 WRITE (6,14) (PRINT(K), K = 1,I)
 71          WRITE (7,19) (PRINT(K),K=1,I)
 76          GO TO 2
 77       13 WRITE (6,14) (PRINT(K), K = 1,L)
104       14 FORMAT (1H ,130A1)
105          WRITE (7,19) (PRINT(K),K=1,L)
112          IF (L.LT.NUM) GO TO 15
115          J = 1
116          W = 1
117          GO TO 17
120       15 V = 0
```

```
571            BAXENDA                        FORTRAN SOURCE LIST HEAD
    ISN        SOURCE STATEMENT

    121           L = L+1
    122           DO 16 N = L,NUM
    123           V = V+1
    124           PRINT(V) = PRINT(N)
    125        16 CONTINUE
    127           J = V+1
    130           W = J
    131        17 IF (M .EQ. 72) GO TO 3
    134           GO TO 7
          C       CHECK FOR AN -END OF ARTICLES- SIGNAL CARD.
    135        18 IF ((RECORD(2).EQ.EX).AND.(RECORD(3).EQ.EX)) CALL EXIT
    140           RETURN
    141        19 FORMAT (132A1)
    142           END
```

```
$IBFTC INDEXT
      SUBROUTINE INDEXT
C     THIS SUBROUTINE CONVERTS THE FIRST TWO CHARACTERS OF A WORD
C     INTO AN INDEX BETWEEN 1 AND 4096.
      COMMON KWORD(14)
      COMMON /INDX/ INDEX
C      GENERATE TRANSFER MATRIX ADDRESS INDEX.
      IF (KWORD(1).LT.0) GO TO 9
      INDEX=KWORD(1)/2**24
      GO TO 10
    9 INDEX=2**11-KWORD(1)/2**24
   10 INDEX=INDEX+1
      RETURN
      END
```

```
L571                   BAXEND                  FORTRAN SOURCE LIST
     ISN       SOURCE STATEMENT

       C $IBFTC
         C CARD-TO-TAPE CONVERSION PROGRAM.
       1         DIMENSION DICT(14)
       2         DATA SIGNAL/6HSIGNAL/
       3         INTEGER DICT,SIGNAL
       4         WRITE (6,4)
       5       4 FORMAT (42H0THE FOLLOWING ARE NEW DICTIONARY ENTRIES.)
       6       1 READ (5,2) DICT
      10       2 FORMAT (14A6)
      11         IF (DICT(1).EQ.SIGNAL) GO TO 3
      14         WRITE (12,2) DICT
      15         WRITE (6,5) DICT
      16       5 FORMAT (1H ,5X,14A6)
      17         GO TO 1
      20       3 REWIND 12
      21         WRITE (6,6)
      22       6 FORMAT (19H END OF INPUT DATA.)
      23         CALL EXIT
      24         END
```

```
XENDA                IBMAP ASSEMBLY CCNCOD                     02/02/68      PAGE  2
ONARY

     $TEXT   CONCOD                                                          CONC000

     * THIS SUBROUTINE CONVERTS A BINARY NUMBER INTO A6 FORMAT FOR PRINTING.
     * THE CALL FORMAT IS ...CALL(X), WHERE X IS THE INTEGER VARIABLE TO
     *  BE CONVERTED.
             ENTRY   CONCOD
     CONCOD  SAVE    1,2,4            SAVE XR 1,2, AND 4
10000        CLA*    3,4              PICK UP THE BINARY NUMBER.
10001        LDQ     ZERO
10001        VDP     TABLE+5,,4
10001        VDP     TABLE+4,,6


10001        VDP     TABLE+3,,6
10001        VDP     TABLE+2,,6
10001        VDP     TABLE+1,,6
10001        VDP     TABLE,,6
10000        STQ*    3,4              RETURN CODED NUMBER TO MAIN PROGRAM.
             RETURN  CONCOD           RETURN TO THE MAIN PROGRAM
10000 TABLE  OCT     200000000000
10000        OCT     024000000000
10000        OCT     003100000000
10000        OCT     000372000000
10000        OCT     000047040000
10000        OCT     000006065000
10000 ZERO   OCT     000000000000
             EXTERN     S.SLOC   GENERATED
01111        END

 CONCOD
```

```
31571                  BAXENDA                       FORTRAN SOURCE LIST
        ISN         SOURCE STATEMENT

          0 $IBFTC OUTWRD  NODECK
          1        SUBROUTINE WRDPUT(LOC,J)
          2        DATA IBLANK/1H /
          3        COMMON /LIN/ LINE(132),L,LM,IDICT(3000)
          4        COMMON KW(14),NW,IWRD(72),LNGTH
            C GET THE LENGTH OF THE WORD TO BE PLACED IN THE PRINT LINE.
          5        NW = IDICT(LOC-2)-(LOC+2)
            C EXTRACT THE WORD FROM THE DICTIONARY.
          6        DO 1 I = 1, NW
          7        II = LOC+I-1
         10      1 KW(I) = IDICT(II)
            C CONVERT THE WORD FROM A6 FORMAT TO A1.
         12        CALL DECOMP
            C SEE IF WORD WILL FIT INTO PRESENT LINE.
         13        IF (L + LNGTH + J - LM) 4,4,2
            C SINCE LINE IS FULL, PRINT IT.
         14      2 WRITE (6,3) (LINE(I),I=1,L)
         21      3 FORMAT (1H ,132A1)
            C REINITIALIZE FOR A NEW LINE.
         22        LINE(1) = IBLANK
         23        L = 1
            C PLACE THE WORD IN THE LINE.
         24      4 DO 5 I = 1,LNGTH
         25        L = L+1
         26      5 LINE(L) = IWRD(I)
         30        RETURN
         31        END
```

9. Operating Instructions

OPERATING INSTRUCTIONS FOR AUTOMATIC INDEXING PROGRAM

Assumes processed dictionary tape on B3. Reads parameter card for width of lines in heading and article. Reads article cards until end of data signal ZZZ in columns 1 - 3.

CONTROL AND COMMENT CARDS

```
$IBFTC EDIT    NODECK
$JOB           31571                     BAXENDALE
$*TAPE INPUT  237 ON B-3
$PAUSE         MOUNT TAPES
$IBJOB         NODECK
$FILE          -FTC02.-,U07,UC7,BLOCK=4097,SINGLE,REEL,TYPE3,LRL=4096,
$ETC           RCT=001,ERR=RERRX.,EOF=REOFX.,EOR=REORX.
```

Automatic indexing program deck.
END

Subroutines follow automatic indexing program deck

$ENTRY EDIT

65 4 2

First parameter card, columns 1-3 is the line width used by HEAD subroutine

Column 5 codes	1 - Print partial comparisons and large words 2 - Print partial comparisons only 3 - Print large words only 4 - Do not list partial comparisons or large words
Column 7 codes	1 - Non-medical terms saved 2 - Non-medical terms not saved

Data consisting of key-punched articles follow.

See Fig. 6 for the arrangement of sentinel cards and articles.

X - 'End of data' signal for HEAD subroutine.

ZZZ - 'End of data' for EDIT

XXX - Means end of all data -   end of program card for HEAD

```
$IBSYS

$PAUSE         UNLOAD, SAVE TAPE CN B-3.

$IBSYS
TIME 320   021468001432021468C01452
CARD 540   021468000308
PAPR 590   021468000037
```

OPERATING INSTRUCTIONS FOR DICTIONARY PROCESSING PROGRAM

Set up tapes on B3 and B4 for output
Set up tapes on B2, B5, B6, B7 as scratch tapes to be used by the sort and merge program

CONTROL AND COMMENT CARDS

```
$JOB           31571                    BAXENDALE
$*TAPE OUTPUT SCRATCH ON B2,B3,B4,B5,B6,B7.
$PAUSE         MOUNT TAPES.
$IBJOB         NODECK
$IBFTC
```

Card to Tape Conversion Source Deck
(This can be replaced by Object Deck)
Card records input - magnetic tape output on B3
$ENTRY
DATA CONSISTING OF
UNSORTED DICTIONARY CARDS
.
.
.
SIGNAL - End of data card
$IBSYS - Return to system

```
31571                    BAXENC               GENERALIZEC SORTING SYSTEM
```

Generalized sort routine: Input on B3 sorted output on B4

```
$IBSYS
$IBSRT          NCTYPE
SCRT MCNITCR NCW LCACING ECIT PHASE.
$* CICTICNARY SCRT
 FILE,INPUT/1,BLCCKSIZE/14,REWINC
 FILE,CLTPLT,BLCCKSIZE/14,SERIAL/2,REWINC
 RECCRC,TYPE/F,LENGTH/14,FIELC/60C
 SCRT,FILE/1,CRCER/2,FIELC/1A
 SYSTEM,INPLT/S.SUC7,MERGE/B,CUTPUT/S.SLC8
 CPTICN,NCCKPT
 ENC
SYSTEM-MERGE-SAME CHANNEL SPECIFIEC TWICE
INPUT LNITS  B3
MERGE LNITS  B2,B5
MERGE LNITS  B6,B7
CUTPLT LNITS B4
SCRT MCNITCR NCW LCACING PHASE CNE - INTERNAL SCRT.
INTERNAL SCRT CCMPLETEC
TCTAL RECCRCS PRCCESSEC           392
SCRT MCNITCR NCW LCACING PHASE THREE - FINAL MERGE.
1C3C1-CLTPLT CN        B4
RECCRCS MERGEC --           392
FINAL MERGE PHASE CCMPLETEC.
SCRT MCNITCR RETURNING TC IBSYS VIA S.SRET
```

DICTIONARY PROCESSING AND CREATION OF TRANSFER MATRIX

Process sorted tape on B4 giving condensed dictionary with transfer address matrix and packages output on B3.

```
$IBJOB

$FILE          -FTC02.-,LO7,LO7,BLOCK=4097,SINGLE,REEL,TYPE3,LRL=4096

$ETC           RCT=C01,ERR=RERRX.,ECF=RECFX.,ECR=RECRX.

$IBFTC DICTDO
```

Dictionary processing source program deck

10. Rules for Keypunching Articles

1. Text to be punched in columns 1-72.

2. Text will be punched continuously; words unfinished in column 72 will be continued in column 1 of the next card.

3. Columns 73-80 are reserved for article identification. The ID will contain a sequential number in column 78, 79, 80. Numbers from 001 to 999.

4. One space between words.

5. A comma is placed in the column next to the last letter of the word preceding it and a space follows the comma.

6. A period at the end of a sentence is placed next to the last letter of the last word of the sentence and is followed by two spaces.

7. An abbreviation period within a sentence is followed by one space.

8. All Roman letters, upper and lower case, italics are treated in the same way. For example, all kinds of "b" are keypunched B.

9. Greek letters - where Greek letters appear replace with English word name of letter. For example: α -Naphthyl keypunched as: ALPHA-NAPHTHYL

10. Card containing END OF ARTICLE in columns 1-14 is placed at end of card deck containing the keypunched article.

11. List the cards on 407 for checking purposes.

12. Tables: copy lines of text.

13. Figures, diagrams and pictures. Treat the description or legend as sentence or sentences.

14. Omit formulae, footnotes, page numbers, acknowledgements, references.

15. Bibliographic entry: Name, initials following; title; journal volume; pages; date. Bibliographic entry to be ended by an sign following immediately after last figure of date.

16. Reproduce bibliographic entry, to be associated with summary or abstract.

17. Keypunch abstract or summary with same rules as for article.

18. If a word ends in column 72 the next word must begin in column 2 leaving column 1 blank.